(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（Ａ）**　(11)特許出願公開番号
特開2003－44093
（P2003－44093A）
(43)公開日　平成15年２月14日(2003.2.14)

(51)Int.Cl.[7]	識別記号	ＦＩ		テーマコード（参考）
Ｇ１０Ｌ　15/28		Ｇ１０Ｌ　3/00	５７１Ｚ	５Ｄ０１５
15/00			５５１Ａ	
15/22			５５１Ｐ	
			５６１Ｈ	
			５７１Ｕ	

審査請求　未請求　請求項の数20　ＯＬ　（全 49 頁）　最終頁に続く

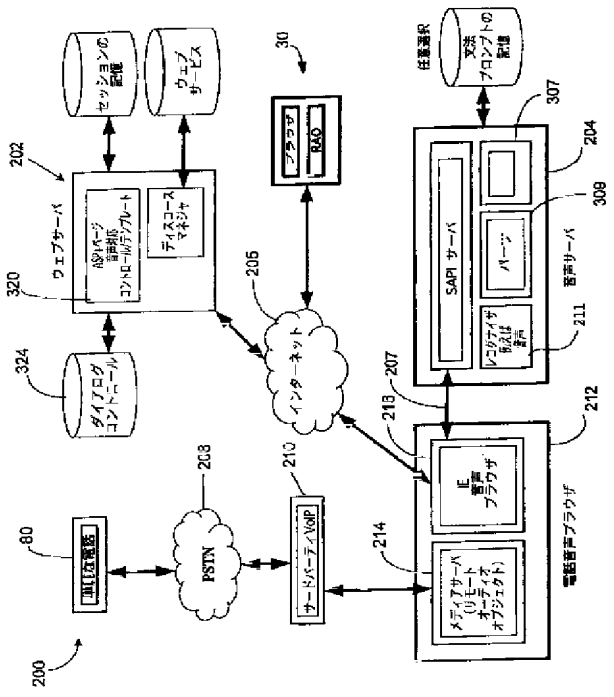
(21)出願番号　特願2002－132052(P2002－132052)
(22)出願日　平成14年５月７日(2002.5.7)
(31)優先権主張番号　６０／２８９，０４１
(32)優先日　平成13年５月４日(2001.5.4)
(33)優先権主張国　米国（ＵＳ）
(31)優先権主張番号　０９／９６０，２２９
(32)優先日　平成13年９月20日(2001.9.20)
(33)優先権主張国　米国（ＵＳ）
(31)優先権主張番号　１０／１１７，１４１
(32)優先日　平成14年４月５日(2002.4.5)
(33)優先権主張国　米国（ＵＳ）

(71)出願人　391056933
マイクロソフト　コーポレイション
ＭＩＣＲＯＳＯＦＴ　ＣＯＲＰＯＲＡＴＩＯＮ
アメリカ合衆国　ワシントン州　98052－6399　レッドモンド　ワン　マイクロソフト　ウェイ　（番地なし）
(74)代理人　10007/481
弁理士　谷　義一　（外２名）

最終頁に続く

(54)【発明の名称】　ウェブ対応音声認識用サーバの方法および記録媒体

(57)【要約】
【課題】　インターネットなどのサーバ／クライアントアーキテクチャで音声認識を提供するのに使用されるウェブ対応音声認識用サーバに、統一したアーキテクチャを持たせる。
【解決手段】　クライアント／サーバシステムのクライアントデバイスで実行するためのマークアップ言語は、非表示式、音声入力ベースのクライアントデバイス８０と、各クライアントデバイスと対話するウェブサーバ２０２用のマルチモーダルベースのクライアント３０とにおける、認識に関連するイベント、ＧＵＩイベント、および電話イベントのうち少なくとも１つを、各クライアントデバイスと対話するウェブサーバ２０２のために統一する命令を含む。クライアントデバイスに提供された入力データを示すデータと、認識に使用する文法の指示とを受信する認識サーバ２０４も提供される。



【特許請求の範囲】
【請求項１】　実施されたときに、コンピュータに情報の処理を行わせるコンピュータ可読命令を含むコンピュータ読み取り可能な記録媒体であって、前記命令は、
クライアントデバイスにおける入力を表すデータと、認識を行うために、前記入力を表すデータに使用する文法の指示とを、ネットワークを介して受信するステップと、
前記入力を表すデータについての認識結果を表すデータを、前記ネットワーク上の遠隔位置に送信するステップとを備えたことを特徴とするコンピュータ読み取り可能な記録媒体。
【請求項２】　前記指示は、前記文法の位置へのリファレンスを提供することを特徴とする請求項１に記載のコンピュータ読み取り可能な記録媒体。
【請求項３】　前記指示は、認識用の言語へのリファレンスを含むことを特徴とする請求項１に記載のコンピュータ読み取り可能な記録媒体。
【請求項４】　マークアップ言語は、ＨＴＭＬ、ＸＨＴＭＬ、ｃＨＴＭＬ、ＸＭＬ、およびＷＭＬの１つを含むことを特徴とする請求項１に記載のコンピュータ読み取り可能な記録媒体。
【請求項５】　マークアップ言語は、スクリプティング言語を含むことを特徴とする請求項１に記載のコンピュータ読み取り可能な記録媒体。
【請求項６】　マークアップ言語は、同期化マルチメディアマークアップ言語を含むことを特徴とする請求項１に記載のコンピュータ読み取り可能な記録媒体。
【請求項７】　レコグナイザは音声レコグナイザを含み、前記文法は音声認識に関連することを特徴とする請求項１に記載のコンピュータ読み取り可能な記録媒体。
【請求項８】　レコグナイザは手書きレコグナイザを含み、前記文法は手書き認識に関連することを特徴とする請求項１に記載のコンピュータ読み取り可能な記録媒体。
【請求項９】　レコグナイザはジェスチャレコグナイザを含み、前記文法はジェスチャ認識に関連することを特徴とする請求項１に記載のコンピュータ読み取り可能な記録媒体。
【請求項１０】　レコグナイザは視覚レコグナイザを含み、前記文法は視覚認識に関連することを特徴とする請求項１に記載のコンピュータ読み取り可能な記録媒体。
【請求項１１】　クライアント／サーバネットワークにおける音声認識の方法であって、
入力音声を表すデータと、認識を行うために、前記入力を表すデータに使用する文法の指示とを、ネットワークを介して受信するステップと、
レコグナイザとともに前記文法を使用して前記データを処理し、認識結果を得るステップと、
前記入力を表すデータについての前記認識結果を、前記ネットワーク上の遠隔位置に送信するステップとを備えたことを特徴とする方法。
【請求項１２】　前記指示は、前記文法の位置へのリファレンスを提供することを特徴とする請求項１１に記載の方法。
【請求項１３】　前記指示は、認識用の言語へのリファレンスを含むことを特徴とする請求項１１に記載の方法。
【請求項１４】　前記遠隔位置にプロンプトを提供するステップをさらに備えたことを特徴とする請求項１１に記載の方法。
【請求項１５】　プロンプトを提供するステップは、テキストデータを音声データに変換するステップと、該音声データを前記遠隔位置に提供するステップとを含むことを特徴とする請求項１４に記載の方法。
【請求項１６】　クライアント／サーバシステム中のクライアントデバイスで実行するためのマークアップ言語を有するコンピュータ読み取り可能な記録媒体であって、
前記マークアップ言語は、非表示式、音声入力ベースのクライアントデバイス、およびマルチモーダルベースのクライアントにおける、認識関連イベント、ＧＵＩイベント、および電話イベントのうち少なくとも１つを、前記クライアントデバイスの各々と対話するウェブサーバのために、統一する命令を備えたことを特徴とするコンピュータ読み取り可能な記録媒体。
【請求項１７】　前記マークアップ言語は、ＨＴＭＬ、ＸＨＴＭＬ、ｃＨＴＭＬ、ＸＭＬ、およびＷＭＬの１つを含むことを特徴とする請求項１６に記載のコンピュータ読み取り可能な記録媒体。
【請求項１８】　前記マークアップ言語は、スクリプティング言語を含むことを特徴とする請求項１６に記載のコンピュータ読み取り可能な記録媒体。
【請求項１９】　前記マークアップ言語は、同期化マルチメディアマークアップ言語を含むことを特徴とする請求項１６に記載のコンピュータ読み取り可能な記録媒体。
【請求項２０】　前記マークアップ言語はスクリプティングを模倣することを特徴とする請求項２１に記載の方法。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、インターネットなどのワイドエリアネットワークを介した情報のアクセスに関する。より詳細には、本発明は、各種の方法を使用してクライアント側で情報およびコントロールを入力することを可能にするウェブ対応認識に関する。
【０００２】
【従来の技術】人々が、個人情報マネジャ（ＰＩＭ）、デバイス、および携帯電話のような小型のコンピューテ

ィングデバイスを日常活動で使用する頻度は増す一方である。現在では、こうしたデバイスを作動させるのに使用されるマイクロプロセッサに利用できる処理能力が増大したことにより、これらデバイスの機能性が高まっており、場合によっては機能を一体化している。例えば現在、携帯電話の多くは、アドレス、電話番号などの個人情報の記憶に使用できるだけでなく、インターネットのアクセスおよびブラウズにも使用することができる。

【０００３】こうしたコンピューティングデバイスをインターネットブラウズに使用し、あるいは他のサーバ／クライアントアーキテクチャで使用することから、情報をコンピューティングデバイスに入力することが必要となる。不都合なのは、携行を容易にするためにこうしたデバイスを可能な限り小さくしたいという要求があり、利用可能なコンピューティングデバイス筐体の表面面積が限られているために、アルファベットの全文字を個別のボタンとして備える従来型のキーボードが通例は不可能であることである。

【０００４】最近、ＶｏｉｃｅＸＭＬ（音声拡張可能マークアップ言語）の使用によるなどの音声ポータルが進歩し、電話だけを使用してインターネットコンテンツにアクセスすることが可能になっている。このアーキテクチャでは、ドキュメントサーバ（例えばウェブサーバ）が、ＶｏｉｃｅＸＭＬインタープリタを通じてクライアントからの要求を処理する。ウェブサーバはそれに応答してＶｏｉｃｅＸＭＬドキュメントを生成することができ、このドキュメントはＶｏｉｃｅＸＭＬインタープリタによって処理し、ユーザに対して音声としてレンダリングされる。ユーザは、音声認識を通じて音声コマンドを使用することにより、ウェブをナビゲートすることができる。

【０００５】ＶｏｉｃｅＸＭＬは、フロー制御タグを用いるマークアップ言語であるが、フロー制御は、イベンティング（ｅｖｅｎｔｉｎｇ）および個別のスクリプトを含むＨＴＭＬ（ハイパーテキストマークアップ言語）のフロー制御モデルには従わない。ＶｏｉｃｅＸＭＬは一般に、電話ベースの音声のみの対話に特に適したフォーム解釈アルゴリズムを含むが、このアルゴリズムでは通例、ユーザから得られる情報をシステムまたはアプリケーションによって制御する。グラフィカルユーザインタフェースも提供し、クライアントーサーバ関係で利用することのできるアプリケーションにＶｏｉｃｅＸＭＬを直接組み込むには、開発者は、２つの形態のウェブオーサリングを習得する必要がある。すなわち、ＶｏｉｃｅＸＭＬのオーサリングと、ＨＴＭＬ（など）を使用したオーサリングであるが、これらはそれぞれ異なるフロー制御モデルに従っている。

【０００６】

【発明が解決しようとする課題】したがって、インターネットなどのサーバ／クライアントアーキテクチャで音声認識を提供するのに使用されるアーキテクチャ、またはその部分、および方法に改良を加えることが現在必要とされている。音声認識用のオーサリングツールは、ＰＩＭ、電話などの小型のコンピューティングデバイスに容易に適合できなければならない。前述の不利点の１つ、いくつか、またはすべてに対処するウェブオーサリングのアーキテクチャまたは方法が特に必要とされる。

【０００７】本発明は、このような課題に鑑みてなされたもので、その目的とするところは、インターネットなどのサーバ／クライアントアーキテクチャで音声認識を提供するのに使用される、統一したアーキテクチャを備えたウェブ対応音声認識用サーバの方法および記録媒体を提供することにある。

【０００８】

【課題を解決するための手段】データ処理用のサーバ／クライアントシステムは、リモートにアクセスできる情報を含んだウェブサーバを有するネットワークを含む。クライアントデバイスは、マイクロフォンなどの入力装置と、スピーカまたはディスプレイなどのレンダリング構成要素を含む。クライアントデバイスは、ウェブサーバから情報を入手して、その情報に含まれるフィールドと関連付けられた入力データを記録するように構成する。クライアントデバイスは、認識に使用する文法の指示とともに入力データを遠隔位置に送信するように適合する。

【０００９】本発明の一態様として、認識サーバは入力データおよび文法の指示を受け取る。認識サーバは、何が入力されたかを示すデータをクライアントおよびウェブサーバの少なくとも１つに戻す。

【００１０】本発明の第２の態様として、クライアント／サーバシステム中のクライアントデバイスで実行するマークアップ言語は、各クライアントデバイスと対話するウェブサーバのために、非表示式の音声入力ベースのクライアントデバイスとマルチモーダルベースのクライアントにおける、認識に関連するイベント、ＧＵＩイベント、および電話イベントのうち少なくとも１つを統一する命令を含む。

【００１１】

【発明の実施の形態】ウェブベース認識のアーキテクチャおよびその実施方法を説明する前に、このアーキテクチャで機能することが可能なコンピューティングデバイスについて全般的に説明しておくと有用であろう。本明細書で図１を参照すると、データ管理デバイス（ＰＩＭ、ＰＤＡなど）の例示的形態が３０に表されている。ただし、本発明は、下記で論じるこの他のコンピューティングデバイス、特に入力ボタンなどを装備するには表面積が限られたコンピューティングデバイスを使用して実施することも企図している。例えば、電話および／またはデータ管理デバイスも、本発明から利益を受けることができる。このようなデバイスは、既存の携帯個人情

報管理デバイスおよびその他の携帯電子デバイスと比較して高いユーティリティを備え、そのデバイスの諸機能とコンパクトなサイズにより、ユーザがデバイスを常に携行することを促すと思われる。したがって、本明細書に記載するアーキテクチャの範囲は、本明細書に記載する例示的なデータ管理デバイスまたはＰＩＭデバイス、電話機、またはコンピュータの開示によっては制限しないものとする。

【００１２】データ管理モバイルデバイス３０の例示的な形態を図１に示す。モバイルデバイス３０は筐体３２を含み、ディスプレイ３４を含むユーザインタフェースを有する。ユーザインタフェースには、スタイラス３３と合わせて接触感知式の表示画面を使用する。スタイラス３３は、指定された座標でディスプレイ３４を押す、またはディスプレイ３４に接触して、フィールドを選択し、カーソルの開始位置を選択的に移動するのに使用し、あるいはジェスチャや手書きなどによる他の方法でコマンド情報を提供するのに使用する。これに代えて、あるいはこれに加えて、ナビゲーション用に１つまたは複数のボタン３５ａ、３５ｂ、３５ｃをデバイス３０上に含むことができる。さらに、回転ホイール、ローラなどの他の入力機構も提供することができる。ただし、本発明は、これらの形態の入力機構によっては制限しないことに留意されたい。例えば、この他の形態の入力には、コンピュータビジョン（ｖｉｓｉｏｎ）を用いるなどの視覚的な入力を含むことができる。

【００１３】次いで図２を参照すると、モバイルデバイス３０を構成する機能構成要素をブロック図で示している。中央演算処理装置（ＣＰＵ）５０は、ソフトウェア制御機能を実施する。ＣＰＵ５０はディスプレイ３４に結合され、制御ソフトウェアに従って生成されるテキストおよびグラフィックアイコンが、ディスプレイ３４に表示される。スピーカ４３を、通例はデジタルからアナログに変換する変換器５９とともにＣＰＵ５０に結合し、音声による出力を提供することができる。ユーザがモバイルデバイス３０にダウンロードまたは入力したデータは、ＣＰＵ５０と双方向に結合した不揮発性の読み出し／書き込みランダムアクセスメモリ記憶装置５４に記憶する。ランダムアクセスメモリ（ＲＡＭ）５４は、ＣＰＵ５０が実行する命令の揮発性の記憶、およびレジスタ値など一時的なデータの記憶を提供する。構成オプションや他の変数のデフォルト値は、読み出し専用メモリ（ＲＯＭ）５８に記憶する。ＲＯＭ５８は、モバイル３０の基本機能、およびその他のオペレーティングシステムカーネル機能（例えばソフトウェアコンポーネントをＲＡＭ５４にロードするなど）を制御する、デバイス用のオペレーティングシステムソフトウェアの記憶にも使用することができる。

【００１４】ＲＡＭ５４は、アプリケーションプログラムの記憶に使用するＰＣのハードドライブ機能と同様の方式で、コードの記憶機構としても機能する。不揮発性メモリをコードの記憶に使用しているが、コードは代わりに、コードの実行には使用されない揮発性メモリに記憶することも可能であることに留意されたい。

【００１５】無線信号は、ＣＰＵ５０に結合された無線トランシーバ５２を通じて、モバイルデバイスによって送信／受信することができる。所望の場合には、コンピュータ（例えばデスクトップコンピュータ）から、あるいは配線式ネットワークから直接データをダウンロードするために、任意選択の通信インタフェース６０を提供することもできる。したがって、インタフェース６０は、例えば赤外線リンク、モデム、ネットワークカードなど、様々な通信装置の形態を備えることができる。

【００１６】モバイルデバイス３０は、マイクロフォン２９、アナログ／デジタル（Ａ／Ｄ）変換器３７、および記憶装置５４に記憶された任意選択の認識プログラム（音声、ＤＴＭＦ、手書き、ジェスチャ、またはコンピュータ画像）を含む。一例として、デバイス３０のユーザからの音声による情報、命令、またはコマンドに応答して、マイクロフォン２９が音声信号を提供し、それをＡ／Ｄ変換器３７でデジタル化する。音声認識プログラムは、デジタル化した音声信号に正規化および／または特徴抽出機能を行って、中間の音声認識結果を得る。無線トランシーバ５２または通信インタフェース６０を使用して、下記で説明し、図５のアーキテクチャに表すリモートの認識サーバ２０４に音声データを送信する。その後認識結果をモバイルデバイス３０に戻して、そこでレンダリング（例えば視覚的かつ／または可聴的に）を行い、最終的にウェブサーバ２０２（図５）に送信するが、本明細書でウェブサーバ２０２とモバイルデバイス３０はクライアント／サーバ関係で動作している。これと同様の処理を、他の形態の入力にも使用することができる。例えば、手書き入力を、デバイス３０での前処理により、または前処理によらずにデジタル化することができる。音声データと同様に、この形態の入力も認識のために認識サーバ２０４に送信することができ、認識結果が、デバイス３０および／またはウェブサーバ２０２の少なくともどちらかに戻される。同様に、ＤＴＭＦデータ、ジェスチャデータ、および視覚データも同じように処理することができる。入力形態に応じて、デバイス３０（および下記で説明する他の形態のクライアント）は、カメラや視覚入力など必要なハードウェアを含む。

【００１７】図３は、携帯電話８０の一例示的実施形態の平面図である。電話機８０は、ディスプレイ８２およびキーパッド８４を含む。一般に、図２のブロック図は図３の電話機にも該当するが、他機能を行うために必須の追加回路が必要になることもある。例えば、図３の実施形態には、電話としての動作に必須のトランシーバが必要になるが、このような回路は本発明には関連しない。

【００１８】上記の携帯式またはモバイル型のコンピューティングデバイス以外にも、本発明は、一般的なデスクトップコンピュータなど数多くの他のコンピューティングデバイスにも使用できることが理解されよう。例えば、身体能力が限られたユーザにとって完全な英数文字キーボードなど他の従来型の入力装置の操作が困難である場合に、本発明は、そのようなユーザがコンピュータまたは他のコンピューティングデバイスにテキストを入力することを可能にする。

【００１９】本発明はまた、数多くの他の汎用または特殊目的のコンピューティングシステム、環境、または構成での動作が可能である。本発明とともに使用するのに適した周知のコンピューティングシステム、環境、および／または構成の例には、従来型の（ｒｅｇｕｌａｒ）電話（画面を備えない）、パーソナルコンピュータ、サーバコンピュータ、携帯用デバイスまたはラップトップデバイス、マルチプロセッサシステム、マイクロプロセッサベースのシステム、セットトップボックス、プログラマブル家庭用電化製品、ネットワークＰＣ、ミニコンピュータ、メインフレームコンピュータ、上記のシステムまたはデバイスなどのうち任意のものを含む分散型コンピューティング環境が含まれるがこれらに限定するものではない。

【００２０】以下は、図４に示す汎用コンピュータ１２０の簡単な説明である。ただし、この場合もコンピュータ１２０は、適切なコンピューティング環境の一例に過ぎず、本発明の使用または機能性の範囲に関して何らの制限を示唆するものではない。また、コンピュータ１２０は、この図に示す構成要素のいずれか、またはその組合せに関する依存性や要件を有するものとも解釈すべきではない。

【００２１】本発明は、プログラムモジュールなどコンピュータで実行するコンピュータ実行可能命令の一般的な状況で説明することができる。一般に、プログラムモジュールには、特定タスクを実行する、または特定の抽象データタイプを実施する、ルーチン、プログラム、オブジェクト、コンポーネント、データ構造などが含まれる。本発明はまた、通信ネットワークを通じてリンクした遠隔処理装置によってタスクを実行する分散型コンピューティング環境でも実施することができる。分散型コンピューティング環境では、プログラムモジュールは、メモリ記憶装置を含む、ローカルおよびリモートどちらのコンピュータ記憶媒体に置いてもよい。以下で、図面の助けを借りて、プログラムおよびモジュールによって実行するタスクを説明する。当業者は、この説明および図面をプロセッサ実行可能命令として実施することができ、この命令はどの形態のコンピュータ読み取り可能な記録媒体にも書き込むことができる。

【００２２】図４を参照すると、コンピュータ１２０の構成要素には、プロセッサ１４０、システムメモリ１５０、およびシステムメモリを含む各種システム構成要素をプロセッサ１４０に結合するシステムバス１４１が含まれるが、これらに限定しない。システムバス１４１は、メモリバスまたはメモリコントローラ、周辺バス、および各種バスアーキテクチャのうち任意のものを使用したローカルバスを含む数種のバス構造のうち任意のものでよい。このようなアーキテクチャには、例えば、ＩＳＡ（Ｉｎｄｕｓｔｒｙ Ｓｔａｎｄａｒｄ Ａｒｃｈｉｔｅｃｔｕｒｅ）バス、ＵＳＢ（Ｕｎｉｖｅｒｓａｌ Ｓｅｒｉａｌ Ｂｕｓ）、ＭＣＡ（Ｍｉｃｒｏ Ｃｈａｎｎｅｌ Ａｒｃｈｉｔｅｃｔｕｒｅ）バス、ＥＩＳＡ（Ｅｎｈａｎｃｅｄ ＩＳＡ）バス、ＶＥＳＡ（Ｖｉｄｅｏ ＥｌｅｃｔｒｏｎｉｃｓＳｔａｎｄａｒｄｓ Ａｓｓｏｃｉａｔｉｏｎ）ローカルバス、およびメザニンバスとしても知られるＰＣＩ（Ｐｅｒｉｐｈｅｒａｌ Ｃｏｍｐｏｎｅｎｔ Ｉｎｔｅｒｃｏｎｎｅｃｔ）バスが含まれるがこれらに限定するものではない。コンピュータ１２０は、通例、各種のコンピュータ読み取り可能な記録媒体を含んでいる。コンピュータ読み取り可能な記録媒体は、コンピュータ１２０からアクセスすることができる任意の利用可能な媒体でよく、これには揮発性および不揮発性媒体、リムーバルおよび取外し不能媒体が含まれる。例えば、コンピュータ読み取り可能な記録媒体は、コンピュータ記憶媒体および通信媒体を含むことができるがこれらに限定しない。コンピュータ記憶媒体には、コンピュータ可読命令、データ構造、プログラムモジュール、またはその他のデータなどの情報を記憶するための任意の方法または技術に実施された、揮発性および不揮発性、リムーバルおよび取外し不能媒体が含まれる。コンピュータ記憶媒体には、ＲＡＭ、ＲＯＭ、ＥＥＰＲＯＭ、フラッシュメモリ、または他のメモリ技術、ＣＤ－ＲＯＭ、デジタル多用途ディスク（ＤＶＤ）、またはその他の光ディスク記憶、磁気カセット、磁気テープ、磁気ディスク記憶または他の磁気記憶装置、あるいは所望の情報の記憶に使用することができ、コンピュータ１２０からアクセスすることが可能な任意の他の媒体が含まれるがこれらに限定するものではない。

【００２３】通信媒体は、通例、搬送波または他の搬送機構などの変調データ信号中のコンピュータ可読命令、データ構造、プログラムモジュール、または他のデータを実施し、また任意の情報伝達媒体を含む。用語「変調データ信号」とは、情報を信号中に符号化するような方式で、その特性の１つまたは複数を設定または変更した信号を意味する。例として、通信媒体には、配線式ネットワークまたは直接配線接続などの配線式媒体、および音響、ＦＲ、赤外線、および他の無線媒体などの無線媒体が含まれるが、これらに限定しない。上記の媒体の任意の組合せも、コンピュータ読み取り可能な記録媒体の範囲に含むものとする。

【００２４】システムメモリ１５０は、読み出し専用メモリ（ＲＯＭ）１５１およびランダムアクセスメモリ（ＲＡＭ）１５２などの揮発性および／または不揮発性メモリの形態でコンピュータ読み取り可能な記録媒体を含む。起動時などにコンピュータ１２０中の要素間の情報の転送を助ける基本ルーチンを含んだ基本入出力システム１５３（ＢＩＯＳ）は、通例ＲＯＭ１５１に記憶する。ＲＡＭ１５２は、通例、プロセッサ１４０から即座にアクセスすることができ、かつ／またはプロセッサ１４０が現在操作しているデータおよび／またはプログラムモジュールを含む。例として、図４にはオペレーティングシステム１５４、アプリケーションプログラム１５５、他のプログラムモジュール１５６、およびプログラムデータ１５７を示しているが、これらに限定しない。

【００２５】コンピュータ１２０は、他のリムーバル／取外し不能、揮発性／不揮発性のコンピュータ読み取り可能な記録媒体も含むことができる。図４には、取外し不能、不揮発性の磁気媒体との読み出しまたは書き込みを行うハードディスクドライブ１６１、リムーバル、不揮発性の磁気ディスク１７２との読み出しまたは書き込みを行う磁気ディスクドライブ１７１、およびＣＤ　ＲＯＭや他の光媒体などのリムーバル、不揮発性の光ディスク１７６との読み出しまたは書き込みを行う光ディスクドライブ１７５を示すが、これらは例にすぎない。この例示的動作環境で使用できる、この他のリムーバル／取外し不可能、揮発性／不揮発性のコンピュータ記憶媒体には、磁気テープカセット、フラッシュメモリカード、デジタル多用途ディスク、デジタルビデオテープ、ソリッドステートＲＡＭ、ソリッドステートＲＯＭなどが含まれるがこれらに限定しない。ハードディスクドライブ１６１は、通例、インタフェース１６０など取外し不能のメモリインタフェースを通じてシステムバス１４１に接続し、磁気ディスクドライブ１７１および光ディスクドライブ１７５は通例、インタフェース１７０などのリムーバルメモリインタフェースによってシステムバス１４１に接続する。

【００２６】上記で説明し、図４に示すドライブおよびそれに関連するコンピュータ記憶媒体は、コンピュータ１２０のコンピュータ可読命令、データ構造、プログラムモジュール、およびその他のデータの記憶を提供する。例えば、図４では、ハードディスクドライブ１６１は、オペレーティングシステム１６４、アプリケーションプログラム１６５、他のプログラムモジュール１６６、およびプログラムデータ１６７を記憶するものとして示している。これらのコンポーネントは、オペレーティングシステム１５４、アプリケーションプログラム１５５、他のプログラムモジュール１５６、およびプログラムデータ１５７と同じものでも、異なるものでもよいことに留意されたい。本明細書では、オペレーティングシステム１６４、アプリケーションプログラム１６５、他のプログラムモジュール１６６、およびプログラムデータ１６７が少なくとも異なるコピーであることを示すために、これらに異なる番号を与えている。

【００２７】ユーザは、キーボード１８２、マイクロフォン１８３、およびマウスやトラックボール、タッチパッドなどのポインティングデバイス１８１などの入力装置を通じて、コンピュータ１２０にコマンドおよび情報を入力することができる。この他の入力装置（図示せず）には、ジョイスティック、ゲームパッド、衛星放送受信アンテナ、スキャナなどが含まれる。これらの入力装置およびこの他の入力装置は、多くの場合、システムバスに結合したユーザ入力インタフェース１８０を通じてプロセッサ１４０に接続するが、パラレルポート、ゲームポート、あるいはユニバーサルシリアルバス（ＵＳＢ）など他のインタフェースおよびバス構造によって接続することもできる。モニタ１８４または他種の表示装置も、ビデオインタフェース１８５などのインタフェースを介して、システムバス１４１に接続する。コンピュータは、モニタ以外にも、スピーカ１８７およびプリンタ１８６など他の周辺出力装置も含むことができ、これらは出力周辺インタフェース１８８を通じて接続することができる。

【００２８】コンピュータ１２０は、リモートコンピュータ１９４など１つまたは複数のリモートコンピュータへの論理接続を使用するネットワーク化環境で動作することができる。リモートコンピュータ１９４は、パーソナルコンピュータ、携帯用デバイス、サーバ、ルータ、ネットワークＰＣ、ピアデバイス、または他の一般的なネットワークノードでよく、通例は上記でコンピュータ１２０との関連で説明した要素の多くまたはすべてを含む。図４に示す論理接続には、ローカルエリアネットワーク（ＬＡＮ）１９１およびワイドエリアネットワーク（ＷＡＮ）１９３が含まれるが、この他のネットワークを含んでもよい。このようなネットワーキング環境は、オフィス、企業規模のコンピュータネットワーク、イントラネット、およびインターネットで一般的に見られる。

【００２９】ＬＡＮネットワーキング環境で使用する場合、コンピュータ１２０は、ネットワークインタフェースすなわちアダプタ１９０を通じてＬＡＮ１９１に接続する。ＷＡＮネットワーキング環境で使用する場合、コンピュータ１２０は通例モデム１９２か、またはインターネットなどのＷＡＮ１９３を介して通信を確立するための他の手段を含む。モデム１９２は、内蔵型でも外付け式でもよく、ユーザ入力インタフェース１８０または他の適切な機構を介してシステムバス１４１に接続することができる。ネットワーク環境では、コンピュータ１２０との関連で図示するプログラムモジュール、またはその一部をリモートのメモリ記憶装置に記憶することができる。例として図４に、リモートアプリケーションプ

ログラム１９５をリモートコンピュータ１９４に常駐するものとして示しているが、これに限定しない。図のネットワーク接続は例示的なものであり、コンピュータ間に通信リンクを確立する他の手段を使用してよいことは理解されよう。

【００３０】図５に、本発明で実施することのできるウェブベース認識のアーキテクチャ２００を示す。一般に、ウェブサーバ２０２に記憶された情報には、モバイルデバイス３０（本明細書では、入力の形態に基づき、適宜、表示画面、マイクロフォン、カメラ、タッチセンシティブパネルなどを有する他形態のコンピューティングデバイスをも表す）を通じて、または情報を音声により、またはキーを押すのに応答して電話機８０が生成するトーンを通じて要求する電話機８０を通じてアクセスすることができる。電話機の場合には、ウェブサーバ２０２からの情報を音声のみによりユーザに提供する。

【００３１】より重要なのは、情報をデバイス３０を通じて得るか、または音声認識を用いて電話機８０を通じて得るかに関係なく、単一の認識サーバ２０４がどちらの動作モードもサポートすることができる点でアーキテクチャ２００が統一されていることである。さらに、アーキテクチャ２００は、周知のマークアップ言語（例えばＨＴＭＬ、ＸＨＴＭＬ、ｃＨＴＭＬ、ＸＭＬ、ＷＭＬなど）の拡張を使用して動作する。したがって、ウェブサーバ２０２に記憶された情報には、これらのマークアップ言語で使用される周知のＧＵＩ方式を用いてアクセスすることもできる。周知のマークアップ言語の拡張を使用することにより、ウェブサーバ２０２でのオーサリングが容易になり、現在存在するレガシーアプリケーションも、音声認識を含むように容易に修正することができる。

【００３２】一般に、デバイス３０は、ウェブサーバ２０２が提供するＨＴＭＬページ、スクリプトなどを実行する。一例として、音声（ｖｏｉｃｅ）認識が必要な場合には、デジタル化したオーディオ信号または音声特徴などの音声データ（オーディオ信号は上記のようにデバイス３０で前処理する）を、音声認識中に使用する文法または言語モデルの指示とともに、認識サーバ２０４に提供する。認識サーバ２０４の実施態様は多くの形態をとることが可能であり、そのうちの１つを図示したが、一般にはレコグナイザ２１１を含む。認識の結果は、所望の場合、または適切な場合にはローカルのレンダリングのためにデバイス３０に戻される。認識と、使用する場合には任意のグラフィカルユーザインタフェースとを通じて情報を編集すると、必要な場合には、デバイス３０はその情報をウェブサーバ２０２に送信し、そこでさらに処理を行い、さらにＨＴＭＬページ／スクリプトを受信する。

【００３３】図５に示すように、デバイス３０、ウェブサーバ２０２、および認識サーバ２０４は共通に（ｃｏｍｍｏｎｌｙ）接続されており、また本明細書ではインターネットなどのワイドエリアネットワークであるネットワーク２０５を通じて個別にアドレス指定することができる。したがって、これらの装置はいずれも物理的に相互に近接して配置する必要はない。特に、ウェブサーバ２０２が認識サーバ２０４を含む必要はない。この方式によると、ウェブサーバ２０２におけるオーサリングを、それが行うべきアプリケーションに集中させることができ、オーサ（ａｕｔｈｏｒ）は認識サーバ２０４の複雑性を知る必要がない。認識サーバ２０４は、独自に設計してネットワーク２０５に接続することができ、それによりウェブサーバ２０２でさらに変更を行わなくとも更新および改良することができる。下記で説明するように、ウェブサーバ２０２は、クライアント側のマークアップおよびスクリプトを動的に生成することのできるオーサリング機構も含むことができる。別の実施形態では、実装マシンの能力に応じて、ウェブサーバ２０２、認識サーバ２０４、およびクライアント３０を組み合わせることができる。例えば、クライアントがパーソナルコンピュータなどの汎用コンピュータを含む場合には、クライアントは認識サーバ２０４を含むことができる。同様に、所望の場合には、ウェブサーバ２０２および認識サーバ２０４を単一マシンに組み込むことが可能である。

【００３４】クライアントデバイスに関して、クライアント／サーバシステムで入力データを処理する方法は、クライアントデバイスのユーザから入力データを得るように構成された拡張を有するマークアップ言語ページをサーバから受信することと、クライアントデバイスでマークアップ言語ページを実行することと、入力データ（ユーザから得た音声、ＤＴＭＦ、手書き、ジェスチャ、または画像を表す）およびそれに関連する文法をクライアントからリモートに位置する認識サーバに送信することと、認識サーバからの認識結果をクライアントで受信することとを含む。クライアント／サーバシステムのクライアントデバイスで実行するマークアップ言語を有するコンピュータ読み取り可能な記録媒体を提供することができ、このマークアップ言語は、そのクライアントデバイスで入力される入力データと関連付ける文法を指示する命令を有する。

【００３５】電話機８０を通じたウェブサーバ２０２へのアクセスには、配線式または無線式の電話網２０８への電話機８０の接続が含まれ、この電話網が電話機８０をサードパーティのゲートウェイ２１０に接続する。ゲートウェイ２１０は、電話機８０を電話音声ブラウザ２１２に接続する。電話音声ブラウザ２１２は、電話インタフェースを提供するメディアサーバ２１４と、音声ブラウザ２１６を含む。デバイス３０と同様に、電話音声ブラウザ２１２は、ウェブサーバ２０２からＨＴＭＬページ／スクリプトなどを受信する。より重要なのは、こ

れらのＨＴＭＬページ／スクリプトが、デバイス３０に提供されるＨＴＭＬページ／スクリプトと同様の形態であることである。この方式によると、ウェブサーバ２０２は、デバイス３０と電話機８０を個別にサポートする必要がなく、さらには標準的なＧＵＩクライアントを個別にサポートする必要もない。むしろ、共通のマークアップ言語を使用することができる。さらに、デバイス３０と同様に、電話機８０から送信される可聴信号からの音声認識は、ネットワーク２０５、または例えばＴＣＰ／ＩＰを使用する専用回線２０７を通じて、音声ブラウザ２１６から認識サーバ２０４に提供される。ウェブサーバ２０２、認識サーバ２０４、および電話音声ブラウザ２１２は、図４に示す汎用デスクトップコンピュータなど任意の適切なコンピューティング環境に実施することができる。

【００３６】ただしＤＴＭＦ認識を用いる場合は、この形態の認識は、一般的には認識サーバ２０４ではなくメディアサーバ２１４で行うことに留意されたい。すなわち、メディアサーバでＤＴＭＦ文法を使用することになる。

【００３７】上記で指摘したように、ＨＴＭＬ、ＸＨＴＭＬ、ｃＨＴＭＬ、ＸＭＬ、ＷＭＬなどのマークアップ言語、または他のＳＧＭＬ由来のマークアップを用いるマークアップ言語は、クライアント／サーバアーキテクチャで認識を提供するコントロールおよび／またはオブジェクトを含むことができる。この方式では、オーサが、このようなアーキテクチャで使用される主流のウェブ開発プラットフォームであるこれらのマークアップ言語に、すべてのツールと専門知識を活用することができる。

【００３８】一般に、コントロールおよび／またはオブジェクトには、次の機能の１つまたは複数を含むことができる。レコグナイザの構成、レコグナイザの実行、および／または後処理のためのレコグナイザコントロールおよび／またはオブジェクト；シンセサイザの構成およびプロンプト再生のためのシンセサイザコントロールおよび／またはオブジェクト；入力文法リソースを指定する文法コントロールおよび／またはオブジェクト；および／または、認識結果を処理するためのバインドコントロールおよび／またはオブジェクト。拡張は、軽量のマークアップレイヤになるように設計し、これにより、聴覚、視覚、手書きなどによるインタフェースの能力を既存のマークアップ言語に付加する。したがって、拡張は次のものには依存しない。例えばＨＴＭＬなど拡張が含まれる高レベルページ；例えばテキストから音声へのフォーマットや文法フォーマットなど、言語リソースへのリファレンスにその拡張が使用した低レベルフォーマット；および認識サーバ２０４で使用する認識プラットフォーム、および音声合成プラットフォームの個々の属性。

【００３９】認識に適したコントロールおよび／またはオブジェクトを有するマークアップを説明する前に、本発明でＨＴＭＬマークアップ言語とともに実施する簡単なＧＵＩの例を考察しておくと有用であると思われる。図６を参照すると、簡単なＧＵＩインタフェースは、オンライン販売を完了するためにクレジットカード情報をウェブサーバに提出することを含む。この例では、クレジットカード情報は、Ｖｉｓａ、ＭａｓｔｅｒＣａｒｄ、あるいはＡｍｅｒｉｃａｎ　Ｅｘｐｒｅｓｓなど、使用するクレジットカードの種類を入力するフィールド２５０を含む。第２のフィールド２５２はクレジットカード番号の入力を可能にし、第３のフィールド２５４は有効期限の入力を可能にする。フィールド２５０、２５２、および２５４に入力した情報を送信するための「提出」ボタン２６４が提供される。

【００４０】図７は、クライアントから上述のクレジットカード情報を得るためのＨＴＭＬコードを示している。一般に、このような形態のマークアップ言語で一般的なように、コードは本体部分２６０とスクリプト部分２６２を含む。本体部分２６０は、実行するアクションのタイプ、使用するフォーム、各種の情報フィールド２５０、２５２、および２５４を指定するコードの行を含み、また提出ボタン２６４（図６）用のコードも含む。この例は、イベントサポートと、埋め込まれたスクリプトホスティングも表しており、提出ボタン２６４が起動されると、スクリプト部分２６２で関数「ｖｅｒｉｆｙ」が呼び出され、または実行される。「ｖｅｒｉｆｙ」関数は、各クレジットカード（Ｖｉｓａ、ＭａｓｔｅｒＣａｒｄ、Ａｍｅｒｉｃａｎ　Ｅｘｐｒｅｓｓ）のカード番号の長さが適切な長さであることを確認する。

【００４１】図８は、音声認識を使用してウェブサーバ２０４に提供するクレジットカード情報を得るための、図６と同じＧＵＩを生成するクライアントマークアップを表している。音声認識については下記で図８～１６との関連で説明するが、本明細書で説明する技術は、手書き認識、ジェスチャ認識、および画像認識にも同様に応用できることを理解されたい。

【００４２】一般に、エクステンション（拡張；一般には「タグ」としても知られる）はＸＭＬ要素の小セットであり、関連する属性およびＤＯＭオブジェクトプロパティ、イベント、およびメソッドを含み、ソースマークアップドキュメントと合わせて使用することにより、認識インタフェース、ＤＴＭＦまたは呼制御をソースページに適用する。エクステンションの形式（ｆｏｒｍａｌｉｔｙ）および意味（ｓｅｍａｎｔｉｃｓ）はソースドキュメントの性質に依存しないので、エクステンションは、ＨＴＭＬ、ＸＨＴＭＬ、ｃＨＴＭＬ、ＸＭＬ、ＷＭＬで、あるいは任意の他のＳＧＭＬ由来のマークアップとともに等しく効果的に使用することができる。エクステンションは、階層的にすることが可能な新しい機能オ

ブジェクトまたは要素を提供するドキュメントオブジェクトモデルに従う。各要素については付録中で詳細に説明するが、一般に、要素には属性、プロパティ、メソッド、イベント、および／または他の「子」要素を含むことができる。

【００４３】本明細書で、エクステンションは、ブラウザを実行するデバイスの機能に応じて、異なる２つの「モード」で解釈できることにも留意されたい。第１のモード「オブジェクトモード」では、全機能を利用することができる。アプリケーションによるエクステンションのプログラム上の操作は、そのデバイスのブラウザが使用可能にするどの機構でも実行することができる。これには、例えば、ＸＨＴＭＬブラウザにおけるＪＳｃｒｉｐｔインタープリタや、ＷＭＬブラウザにおけるＷＭＬＳｃｒｉｐｔインタープリタなどがある。この理由から、エクステンションのコアプロパティおよびメソッドの小セットだけを定義すればよく、これらは、デバイスすなわちクライアント側に存在する任意のプログラム機構によって操作される。オブジェクトモードは、イベンティングおよびスクリプティングを提供し、またより多くの機能を提供して、ダイアログのオーサに、音声対話に対するより細かなクライアント側におけるコントロールを与えることができる。本明細書で使用する場合、フルイベントおよびスクリプティングをサポートするブラウザを「アップレベルブラウザ」と呼ぶ。この形のブラウザは、エクステンションのすべての属性、プロパティ、メソッド、およびイベントをサポートする。アップレベルブラウザは、通例、より高い処理能力を持つデバイスで使用される。

【００４４】エクステンションは、「宣言モード」でもサポートすることができる。本明細書で使用する場合、宣言モードで動作するブラウザを「ダウンレベルブラウザ」と呼び、これは完全なイベンティングおよびスクリプティング機能はサポートしない。代わりにこの形のブラウザは、所与のエクステンションの宣言的側面（すなわちコア要素および属性）をサポートするが、ＤＯＭ（ドキュメントオブジェクトモデル）オブジェクトのプロパティ、メソッド、およびイベントのすべてはサポートしない。このモードは専ら宣言構文だけを用い、さらに、ＳＭＩＬ（同期化マルチメディア統合言語）２．０などの宣言マルチメディア同期化および協調機構（同期マークアップ言語）と併せて使用することができる。ダウンレベルブラウザは、通例、処理能力が限られたデバイスで使用される。

【００４５】ここで、特定の入力モードについて論じておきたい。詳細には、音声認識を、少なくともディスプレイと併せて、そして別の実施形態ではポインティングデバイスとも併せて使用して、データ入力フィールドを指定すると特に有用である。具体的には、このモードのデータ入力では、ユーザは一般に、いつフィールドを選択し、それに対応する情報を提供するかを制御することができる。例えば、図６の例では、ユーザはまずフィールド２５２にクレジットカード番号を入力し、次いでフィールド２５０にクレジットカードの種類を入力し、最後にフィールド２５４に有効期限日を入力することができる。同様に、ユーザは、所望の場合にはフィールド２５２に戻り、誤った入力を訂正することもできる。下記で説明するように音声認識と組み合わせると、平易で自然なナビゲーション形態が提供される。本発明で使用する場合、フィールドを自由な形で選択することを可能にする画面表示と、音声認識の両方を使用するこの形の入力を「マルチモーダル」と呼ぶ。

【００４６】再び図８を参照すると、ＨＴＭＬマークアップ言語のコードが示されている。図７に示すＨＴＭＬコードと同様に、このコードも、本体部分２７０およびスクリプト部分２７２を含んでいる。また図７に示すコードと同様に、図８に示すコードは、フォームの位置および実行するアクションのタイプに関する指示を含む。フィールド２５０、２５２、および２５４それぞれへの情報の入力は、各々コード部分２８０、２８２、および２８４によって制御または実行する。初めにコード部分２８０を参照すると、例えばデバイス３０のスタイラス３３を使用してフィールド２５０を選択すると、イベント「ｏｎＣｌｉｃｋ」が開始され、これによりスクリプト部分２７２の関数「ｔａｌｋ」が呼び出されるか、または実行される。このアクションは、一般にフィールド２５０に予想されるデータタイプと関連付けられた、音声認識で使用する文法を起動する。複数の入力技術（例えば音声とペンクリック／ローラ）を使用するこの種の対話を「マルチモーダル」と呼ぶ。

【００４７】図８に例示する音声認識エクステンションは、クライアントのブラウザにおいてデフォルトの視覚表現を有さないことに留意されたい。これは、多くのアプリケーションでは、オーサが、アプリケーション仕様のグラフィック機構をソースページで使用することにより、ページの各種コンポーネントの音声使用可能を知らせることを想定しているためである。それでも、視覚的な表現が望ましい場合には、エクステンションをそのように修正することができる。

【００４８】再び文法を参照すると、この文法は、文脈自由文法、Ｎ文法、ハイブリッド文法などの構文文法であるがこれらに限定しない。（言うまでもなく、それに対応する形態の認識を利用する際には、ＤＴＭＦ文法、手書き文法、ジェスチャ文法、および画像文法を使用する。本明細書で使用する場合、「文法」とは認識を行うための情報を含み、別の実施形態では、例えば特定のフィールドに入力されることが予想される入力に対応する情報を含む。）マークアップ言語の最初のエクステンションを含む新しいコントロール２９０（本明細書では「ｒｅｃｏ」と識別する）は様々な要素を含むが、その

うち２つを図に示す。すなわち文法要素「ｇｒａｍｍａｒ」と「ｂｉｎｄ」要素である。一般に、ウェブサーバ２０２からクライアントにダウンロードするコードと同様に、文法はウェブサーバ２０２を発信元とし、クライアントにダウンロードするか、かつ／または音声処理のためにリモートサーバに転送することができる。文法は次いで、そのキャッシュでローカルで記憶することができる。最終的に、文法は認識に使用するために認識サーバ２０４に提供する。文法要素は、インライン文法、または属性を使用して参照する文法を指定するのに使用する。

【００４９】認識を行った音声、手書き、ジェスチャ、画像などに対応する認識結果を認識サーバ２０４から受け取ると、ｒｅｃｏコントロール２９０の構文を提供してそれに対応する結果を受け取り、それを対応フィールドと関連付けるが、これにはその中のテキストをディスプレイ３４でレンダリングすることを含んでもよい。本明細書に例示する実施形態では、音声認識が終了し、結果をクライアントに送り返すと、クライアントはｒｅｃｏオブジェクトを非活動化して、認識済みのテキストをそれに対応するフィールドと関連付ける。コード部分２８２および２８４もこれと同様に動作し、フィールド２５２および２５４ごとに固有のｒｅｃｏオブジェクトおよび文法を呼び出し、認識されたテキストを受け取ると、それをフィールド２５２および２５４とそれぞれ関連付ける。カード番号フィールド２５２の受信については、関数「ｈａｎｄｌｅ」が、上記で図７との関連で説明したのと同様の方式で、カードの種類からカード番号の長さを確認する。

【００５０】一般に、アーキテクチャ２００およびクライアント側のマークアップ言語と併せた音声認識の使用は、次のように行われる。まず、与える音声と関連付けられたフィールドを指示する。図の実施形態ではスタイラス３３を使用するが、本発明はスタイラス３３の使用に限定するものではなく、ボタン、マウスポインタ、回転ホイールなど任意形態の指示を使用できることは理解されよう。周知のように、視覚的なマークアップ言語を使用して、「ｏｎＣｌｉｃｋ」などそれに対応するイベントを提供することができる。本発明は、音声、手書き、ジェスチャなどのコマンドの開始を指示するのに、「ｏｎＣｌｉｃｋ」イベントの使用だけに限定しない。「ｏｎＳｅｌｅｃｔ」など、任意の利用可能なＧＵＩも同じ目的に使用することができる。一実施形態では、このようなイベンティングは、それに対応する音声の開始および／または終わりの両方を示す役割を果たすので、特に有用である。また、音声の対象とするフィールドは、ユーザの対話を追跡するブラウザ上で実行されるプログラムによっても、ユーザによっても指定できることに留意されたい。

【００５１】ここで注意したいのは、異なる音声認識シナリオには、認識サーバ２０４の異なる振る舞いおよび／または出力が必要となることである。認識プロセスの開始はすべての場合に標準的なものであり、すなわちアップレベルブラウザからの明示的なｓｔａｒｔ（　）の呼び出しであり、あるいはダウンレベルブラウザでは宣言的な＜ｒｅｃｏ＞要素であるが、音声認識を中止する手段は異なる可能性がある。

【００５２】上記の例では、マルチモーダルアプリケーションのユーザは、例えば圧力を感知するディスプレイを軽く叩き、接触状態を保持することにより、デバイスへの入力を制御する。するとブラウザは、例えば「ｐｅｎ－ｕｐ」などのＧＵＩイベントを使用して、認識をいつ中止するかを制御し、その後それに対応する結果を戻す。ただし、電話アプリケーション（下記で説明する）あるいは手を使用せずに済むアプリケーションといった音声のみのシナリオでは、ユーザはブラウザに対する直接的な決定権は一切持たず、認識サーバ２０４またはクライアント３０が、いつ認識を中止して結果を戻すか（通例は、文法中のパスを認識した時点）を決定する責任を負わなければならない。さらに、認識を中止する前に中間の結果を戻す必要があるディクテーションや他のシナリオ（「オープンマイクロフォン」としても知られる）の場合には、明示的な中止機能が必要とされるだけでなく、認識プロセスを中止する前に複数の認識結果をクライアント３０および／またはウェブサーバ２０２に戻す必要もある。

【００５３】一実施形態では、Ｒｅｃｏ要素は、下記の３つの認識モードを区別する「ｍｏｄｅ」属性を含むことができ、これにより認識サーバ２０４に、いつどのように結果を戻すかを命令する。結果を戻すことは、「ｏｎＲｅｃｏ」イベントを提供する、または「ｂｉｎｄ」要素を適宜起動することを意味する。一実施形態では、モードを指定しない場合、デフォルトの認識モードは「自動」にすることができる。

【００５４】図１４は、音声認識の「自動」モードの動作を図式的に表したものである（他の形態の認識にもこれと同様のモード、イベントなどを提供することができる）。スケジュール２８１は、認識サーバ２０４にいつ認識の開始２８３を指示するか、認識サーバ２０４がどこで音声を検出し（２８５）、その音声が終了したこと（２８７）を判定するかを表している。

【００５５】Ｒｅｃｏ要素の各種の属性は、認識サーバ２０４の振る舞いを制御する。属性「ｉｎｉｔｉａｌＴｉｍｅｏｕｔ」２８９は、認識の開始２８３から音声の検出２８５までの間の時間である。この期間を超えると、「ｏｎＳｉｌｅｎｃｅ」イベント２９１が認識サーバ２０４から提供され、認識が中止されたことを知らせる。認識サーバ２０４が、発声が認識不可能であると識別した場合は、「ｏｎＮｏＲｅｃｏ」イベント２９３を発行するが、これも認識を中止したことを示す。

【００５６】認識を中止またはキャンセルすることができる他の属性には、「ｂａｂｂｌｅＴｉｍｅｏｕｔ」属性２９５があるが、これは２８５の音声の検出後に認識サーバ２０４が結果を戻さなければならない期間である。この期間を超えると、エラー発生の有無に応じて異なるイベントが発行される。例えば、例外的に発声が長い場合など、認識サーバ２０４がなおオーディオの処理を行っている場合は、「ｏｎＮｏＲｅｃｏ」属性２９３を発行する。しかし他の何らかの理由で「ｂａｂｂｌｅＴｉｍｅｏｕｔ」属性２９５を超えた場合は、認識エラーの可能性が高くなり、「ｏｎＴｉｍｅｏｕｔ」イベント２９７が発行される。同様に「ｍａｘＴｉｍｅｏｕｔ」属性２９９も提供することができ、これは、認識の開始２８３から結果をクライアント３０に戻すまでの期間である。この期間を超えると、「ｏｎＴｉｍｅｏｕｔ」イベント２９７が発行される。

【００５７】ただし、「ｅｎｄＳｉｌｅｎｃｅ」属性３０１以上の期間を超えた場合、これは認識が完了していることを示唆するが、この場合は認識サーバ２０４が自動的に認識を中止し、その結果を戻す。認識サーバ２０４は、信頼度の測定を実施して、認識結果を戻すべきかどうかを判定できることに留意されたい。信頼度の測定値が閾値を下回る場合は、「ｏｎＮｏＲｅｃｏ」属性２９３を発行し、一方信頼度の測定値が閾値を上回る場合は、「ｏｎＮｏＲｅｃｏ」属性３０３および認識結果を発行する。したがって図１４は、「自動モード」で、明示的なｓｔｏｐ（　）の呼び出しが行われていない状況を表している。

【００５８】図１５は、認識サーバ２０４の「シングルモード」の動作を図式的に表したものである。「自動モード」との関連で上記で説明した属性およびイベントを適用することができ、したがって同じ参照番号で示している。しかし、この動作モードでは、ｓｔｏｐ（　）呼び出し３０５を、スケジュール２８１上に示している。ｓｔｏｐ（　）呼び出し３０５は、ユーザによる「ペンアップ」などのイベントに相当する。この動作モードでは、認識結果を戻すことは、明示的なｓｔｏｐ（　）呼び出し３０５によって制御される。すべての動作モードの場合と同じく、「ｏｎＳｉｌｅｎｃｅ」イベント２９１は、「ｉｎｉｔｉａｌＴｉｍｅｏｕｔ」期間２８９内に音声が検出されない場合に発行されるが、この動作モードでは認識を中止しない。同様に、ｓｔｏｐ（　）呼び出し３０５以前の認識不可能な発声によって生成される「ｏｎＮｏＲｅｃｏ」イベント２９３によっても認識は中止されない。ただし、「ｂａｂｂｌｅＴｉｍｅｏｕｔ」属性２９５または「ｍａｘＴｉｍｅｏｕｔ」属性２９９と関連付けられた期間を超えた場合は、認識を中止する。

【００５９】図１６は、認識サーバ２０４の「複数モード」の動作を図式的に表している。上記で指摘したように、この動作モードは、「オープンマイクロフォン」またはディクテーションのシナリオで使用する。一般に、この動作モードでは、明示的なｓｔｏｐ（　）呼び出し３０５が受け取られるか、または「ｂａｂｂｌｅＴｉｍｅｏｕｔ」性２９５または「ｍａｘＴｉｍｅｏｕｔ」属性２９９に関連付けられた期間を超えるまで、間隔を置いて認識結果を戻す。ただし、「ｏｎＳｉｌｅｎｃｅ」イベント２９１、「ｏｎＲｅｃｏ」イベント３０３、または「ｏｎＮｏＲｅｃｏ」イベント２９３のいずれかが発生すると、これらによって認識は中止されないが、「ｂａｂｂｌｅＴｉｍｅｏｕｔ」期間および「ｍａｘＴｉｍｅｏｕｔ」期間のタイマがリセットされることに留意されたい。

【００６０】一般に、この動作モードでは、ｓｔｏｐ（　）呼び出し３０５が受け取られるまで、認識されるフレーズごとに、「ｏｎＲｅｃｏ」イベント３０３を発行し、結果を戻す。認識不可能な発声のために「ｏｎＳｉｌｅｎｃｅ」イベント２９１が発行された場合は、これらのイベントを報告するが、認識は継続する。

【００６１】上記で触れたように、フィールドに関連付けられた１つまたは複数のｒｅｃｏオブジェクトを起動するが、これには、少なくともどの文法を使用するかについての指示を認識サーバ２０４に提供することが含まれる。この情報は、クライアント３０で記録して認識サーバ２０４に送信した音声データを伴うことができる。上記で指摘したように、音声データは、ユーザが入力した音声に関連づけられたストリーミングデータを含むことができ、あるいは音声認識中に使用する音声の特徴を示す、前処理済みの音声データを含むことができる。別の実施形態では、クライアント側の処理に音声データの正規化も含むことができ、認識サーバ２０４が受け取る音声データが、クライアントごとに比較的均質になるようにする。これにより認識サーバ２０４の音声処理が簡略化され、認識サーバを、クライアントおよび通信経路のタイプにステートレスにすることができるので、認識サーバ２０４のスケーラビリティをより容易にすることができる。

【００６２】認識サーバ２０４から認識結果を受け取ると、その認識結果を対応するフィールドと関連付け、必要な場合はクライアント側で確認またはチェックを行うことができる。現在クライアントがレンダリングしているコードと関連付けられたすべてのフィールドを完了すると、アプリケーション処理のためにその情報をウェブサーバ２０２に送信する。前述の内容から、ウェブサーバ２０２は、認識に適したコードまたはページ／スクリプトをクライアント３０に提供しているが、認識サービスはウェブサーバ２０２によっては行われず、認識サーバ２０４によって行われることが明白であろう。ただし、本発明は、認識サーバ２０４をウェブサーバ２０２とまとめて配置する、または認識サーバ２０４をクライ

アント３０の一部とするような実施を排除するわけではない。すなわち、本明細書で提供するエクステンションは、認識サーバ２０４をウェブサーバ２０２またはクライアント３０と組み合わせた場合でも有用である。これは、エクステンションが、これら構成要素間に単純かつ利便なインタフェースを提供するからである。

【００６３】図８に示す実施形態には示していないが、ｒｅｃｏコントロールは、適切な音声データを認識サーバ２０４に導くためのリモートオーディオオブジェクト（ＲＡＯ）も含むことができる。ＲＡＯをプラグインオブジェクトにすることによる利益は、サウンドインタフェースが異なる可能性が高いことから、異なるデバイスまたはクライアントそれぞれに異なるＲＡＯを可能にすることである。さらに、リモートオーディオオブジェクトにより、複数のｒｅｃｏ要素を同時に起動することが可能になる。

【００６４】図９および１０は、本発明でページ／スクリプトを含むＨＴＭＬとして実施する音声のみによるマークアップ言語を示す。図に明瞭に示すように、このコードも本体部分３００およびスクリプト部分３０２を含んでいる。マークアップ言語の別のエクステンション、すなわちバージインなどの属性を含むプロンプトコントロール３０３がある。ただし、図９および１０の音声のみの実施形態では、音声認識を別の方式で行う。この場合は、プロセス全体を、未入力（ｕｎｆｉｌｌｅｄ）のフィールドを判定し、かつそれに対応するプロンプトおよび新しいオブジェクトを起動するスクリプト関数「ｃｈｅｃｋＦｉｌｌｅｄ」によって制御する。しかし、上記で図８との関連で説明したのと同じコンテクストを使用して文法を起動し、音声データおよび使用する文法の指示を認識サーバ２０４に提供する。同様に、認識サーバ２０４から受け取った出力を、クライアント（この場合は電話音声ブラウザ２１２）のフィールドと関連付ける。

【００６５】一般に音声のみのアプリケーションに固有の他の機能は、音声が認識されなかった際にユーザにそれを知らせることである。図８のようなマルチモーダルのアプリケーションでは、「ｏｎＮｏＲｅｃｏ」は、表示されるフィールドに単にヌル値を入れて、認識が行われなかったことを示すので、それ以上の動作は必要とされない。音声のみの実施形態では、「ｏｎＮｏＲｅｃｏ」３０５は関数「ｍｕｍｂｌｅ」を呼び出し、または実行する。この関数は、単語のフレーズを認識サーバ２０４に転送し、このフレーズは適切なテキストから音声に変換するシステム３０７（図５）を使用して音声に変換される。認識サーバ２０４は、オーディオストリームを電話音声ブラウザ２１２に戻し、次いでユーザが聴くためにそれを電話機８０に送信する。同様に、音声のみのアプリケーションに実施するこの他の波形プロンプトも、必要な場合には認識サーバ２０４によりオーディオストリームに変換する。

【００６６】この例では、関数「ｗｅｌｃｏｍｅ」を介してｗｅｌｃｏｍｅプロンプトを再生すると、関数「ｃｈｅｃｋＦｉｌｌｅｄ」がユーザに各フィールドを指示し、適切な文法を起動する。これには、入力されたフィールドを反復して、その情報が正しいことを確認することが含まれ、また「ｃｏｎｆｉｒｍａｔｉｏｎ」文法の起動が含まれる。この実施形態では、各ｒｅｃｏコントロールは、先の例の本体部分ではなくて、スクリプト部分３０２から開始されることに留意されたい。

【００６７】マークアップ言語は、異なるタイプのクライアントデバイス（例えば、マルチモーダル、および電話機のような非表示式、音声入力ベースのクライアントデバイス）で実行することができ、各クライアントデバイスと対話するウェブサーバのために、認識に関連するイベント、ＧＵＩイベント、および電話イベントのうち少なくとも１つを統一する。これは、ウェブサーバアプリケーションのかなりの部分を、汎用的に、あるいはクライアントデバイスのタイプに依存せずに書くことを可能にするので特に有用である。「ｈａｎｄｌｅ」関数を含む一例を図８、および図９、１０に示す。

【００６８】図９、１０には示していないが、このマークアップ言語には、電話機能をサポートするエクステンションがさらに２つある。すなわち、ＤＴＭＦ（デュアルトーン変調周波）制御と、呼制御の要素またはオブジェクトである。ＤＴＭＦは、ｒｅｃｏコントロールと同様の働きをする。これは、キーパッドストリングからテキスト入力への単純な文法マッピングを指定する。例えば、「１」は食料品部門を意味し、「２」は薬品部門を意味するなどである。一方、呼オブジェクトは、呼の転送や第三者の呼出しのような電話機能を扱う。属性、プロパティ、メソッド、イベントについては付録で詳細に説明する。

【００６９】図１１および１２は、音声のみの動作モードに適したマークアップ言語のさらに別の例を示す。この実施形態では、ユーザは、情報をいつ入力するか、または話すかに関してある程度の制御権を有することができる。言い換えると、このシステムでは、発話を開始させるか、あるいはその他の方法で発話を開始するようにユーザに指示することができるが、ユーザは当初要求されるよりも多くの情報を提供することができる。これは、「混合主導型」の一例である。一般に、この形のダイアログ対話では、ユーザはダイアログの主導権をシステムと分かち合うことができる。上記で触れ、下記で詳細に説明する、ユーザがプロンプトに要求されるよりも多くの情報を提供する例のほかにも、ユーザはその指示がないときにタスクを切り替えることもできる。

【００７０】図１１および１２の例では、「ｄｏ＿ｆｉｅｌｄ」と識別する文法は、文法「ｇ＿ｃａｒｄ＿ｔｙｐｅｓ」、「ｇ＿ｃａｒｄ＿ｎｕｍ」、および「ｇ＿ｅ

ｘｐｉｒｙ＿ｄａｔｅ」と関連付けられた情報を含む。この例では、電話音声ブラウザ２１２は、「ｏｎＲｅｃｏ」として示す認識済みの音声を受け取ると、電話機８０から受け取った音声データと、「ｄｏ＿ｆｉｅｌｄ」文法の使用の指示を認識サーバ２０４に送信し、関数「ｈａｎｄｌｅ」が呼び出され、または実行されるが、これには音声データから認識されたフィールドの一部またはすべての値を関連付けることが含まれる。すなわち、認識サーバ２０４から得る結果は、各フィールドについての指示も含んでいる。この情報は構文解析し、４０５で指定されるバインド規則に従って対応するフィールドと関連付ける。図５に示すように、認識サーバ２０４はパーサ３０９を含むことができる。

【００７１】図７、８、９、１０、１１、および１２から、非常に類似したウェブ開発フレームワークを使用する。データの提示も、これらの各場合で非常に類似している。さらに、データ提示とフロー制御を分離することにより、異なるアプリケーション（システム主導型と混合主導型）間、または異なるモダリティ間（ＧＵＩウェブベース、音声のみ、およびマルチモーダル）での再使用性を最大限にすることができる。また、これにより、電話機がディスプレイおよびデバイス３０と同様の機能を含む場合に、音声のみの動作から電話、そしてマルチモーダル動作への自然な拡張が可能になる。付録Ａでは、以上で説明したコントロールおよびオブジェクトの詳細をさらに提供する。

【００７２】上記で指摘したように、アップレベルブラウザは、上記の例で認識結果を割り当てるために関数「ｈａｎｄｌｅ」を起動するなど、各種のニーズを実行するためにスクリプティングを使用することができる。上記で説明し、付録Ａの２．１．２にさらに説明する実施形態では、「ｂｉｎｄ」要素は認識結果を構文解析し、値を割り当てるが、この「ｂｉｎｄ」要素は「ｒｅｃｏ」要素の下位要素または子要素である。

【００７３】スクリプティングは有用でありうるが、多くの者は、例えばセキュリティ問題などから必ずしも最良のブラウザ実装形態であるとは限らないと見ている。したがって、本発明のさらに別の実施形態または態様では、「ｂｉｎｄ」要素は（「ｒｅｃｏ」同様の）高レベル要素であり、他のより豊富なプロパティとともに提供され、実際、それ自体ではスクリプティングを用いずにスクリプティングを実際に模倣することができる。

【００７４】スクリプティングを用いない場合、あるいは下記で述べる本発明の態様を使用しない場合、高度なダイアログ効果など下記で述べる機能の一部は、ページを再度ウェブサーバ２０２に提出し、そこでアプリケーションロジックを実行して新しいページを生成し、そのページを再びクライアントデバイスに送信することによってのみ実現することができる。本発明のこの態様により、プログラマは、サーバへのラウンドトリップを招く（ｉｎｃｕｒ）ことなく、そのページのオブジェクトのメソッドを起動することができる。

【００７５】上記の実施形態では、「ｂｉｎｄ」要素は、認識結果をフォーム中またはウェブページ中のフィールドに割り当てるための属性「ＴａｒｇｅｔＥｌｅｍｅｎｔ」および「ＴａｒｇｅｔＡｔｔｒｉｂｕｔｅ」しか有さない。別の実施形態では、「ｂｉｎｄ」要素は、オブジェクトメソッドの起動のために加える「ＴａｒｇｅｔＭｅｔｈｏｄ」も含む。「ＴａｒｇｅｔＭｅｔｈｏｄ」の使用および機能は、スクリプティングの模倣にとって非常に重要な技術である。例えば、次の構文を使用して、オブジェクト「ＯＢＪ１」の「Ｘ」メソッドを起動することができる。

```
<bind TargetElement = "OBJ1" TargetMethod = "X"
 ...>
```

ここに示す例はＨＴＭＬ／ＸＨＴＭＬのイベント構文に従っているが、当業者にとっては、＜ｂｉｎｄ＞の使用を一般化して、他のイベンティング機構を使用することは平易であることに留意されたい。他のイベンティング機構には、Ｗ３Ｃドキュメントオブジェクトモデルレベル２またはレベル３のイベンティング規格、ＥＣＭＡ共通言語基盤（ＣＬＩ）イベントモデル、Ｊａｖａ（登録商標）プログラミング言語イベントモデル、Ｗ３Ｃ同期マルチメディア統合言語（ＳＭＩＬ）、および近く登場するＷ３ＣのＸＭＬイベント規格提案が含まれるが、これらに限定するものではない。

【００７６】図１７および１８は、クライアント、特にダウンレベルブラウザで実行可能なマークアップ言語のページである。この例では、音声プロンプトを通じてユーザに希望する飲料を尋ねている。このシステムは次いで、どの飲料が注文されたかを確認する。認識結果に応じて、「ｂｉｎｄ」要素は、宣言した論理を使用して実行を導く。飲料を確認すると、そのフォームをウェブサーバ２０２に再度提出するが、これらにスクリプティングは一切用いない。

【００７７】一般に、図１７および１８のマークアップ例は、データ部分３５０、音声部分３５２、およびユーザインタフェース部分３５４、３５６、および３５８を含む。部分３５４は、全般的な質疑から、ユーザが希望する飲料についての認識結果を受け取り、対話式認識フローを誘導して、クリームや砂糖が必要かどうかについて再度指示を促し、尋ねるか、または注文された飲料を確認する。詳細には、部分３５６は、クリームや砂糖も注文された場合にはその認識結果を受け取る。部分３５８は、飲料の確認についての認識結果を受け取る。部分３６０は、新しいメッセージングオブジェクト「ＳＭＥＸ」を用いる呼制御部分である。「ＳＭＥＸ」については下記でさらに説明する。

【００７８】上記で指摘したように、本発明のこの態様の「ｂｉｎｄ」要素はオブジェクトメソッドの起動を含

み、これは、「ｗｅｌｃｏｍｅ」オブジェクトの「ｓｔａｒｔ」メソッドを３６１で実行する際に「ｗｅｌｃｏｍｅ」プロンプトを再生することにより、図１７および１８の例でユーザ対話を開始する。

【００７９】次いで、３６２で「ａｓｋｅｄ」オブジェクトの「ｓｔａｒｔ」メソッドを実行することにより、ユーザに「ご希望はコーラ、コーヒー、それともオレンジジュースですか？」と尋ねる。次いで、３６３で、認識「ｒｅｃｏ＿ｄｒｉｎｋ」オブジェクトの「ｓｔａｒｔ」メソッドを起動することにより認識を実行する。

【００８０】次いで部分３５４のマークアップを実行するが、ここで認識サーバ２０４が使用する文法は、ＸｐａｔｈステートメントＳ「．／ｄｒｉｎｋ　ｔｙｐｅｓ」によって提供される。この例ではＷ３ＣのＸｐａｔｈ言語を利用しているが、この概念を、他の標準的言語に拡張することは当業者にとって平易であることに留意されたい。他の標準的言語には、Ｗ３ＣによるＸＭＬクエリ言語（ＸＱＬ）を含むが、これに限定するものではない。「ｂｉｎｄ」要素３６４によって明確に示すように、認識サーバ２０４から受け取った認識結果の信頼度スコアが１０未満である場合は、３６６でプロンプトオブジェクト「ｒｅｐｒｏｍｐｔ」を実行し、それに続いてプロンプトオブジェクト「ａｓｋ」を３６８で実行し、この時に認識オブジェクト「ｒｅｃｏ＿ｄｒｉｎｋ」を３７０で再度開始する。戻された認識結果が「ｃｏｆｆｅｅ」で、それが１０を超える信頼度を有する場合、３７２でフィールド「ｄｒｉｎｋ」に認識結果の値を割り当て、３７４でプロンプトオブジェクト「ｃｒｅａｍ＿ｓｕｇａｒ」により、クリームあるいは砂糖を希望するかしないかについてユーザに指示を促す。次いで、３７６で、部分３５６の認識オブジェクト「ｒｅｃｏ＿ｃｒｅａｍ＿ｓｕｇａｒ」を起動する。そうでなく、認識結果が信頼度スコアは１０を超えるがコーヒーでない場合は、３７８でフィールド「ｄｒｉｎｋ」に再度値を割り当てる。認識結果の確認は、プロンプトオブジェクト「ｃｏｎｆｉｒｍ」を実行し、それに続いて部分３５８の認識オブジェクト「ｒｅｃｏ＿ｙｅｓｎｏ」を３８２で起動することにより、３８０で提供する。ユーザが「ｙｅｓ」と答え、その信頼度スコアが１０を超える場合は、３８４でプロンプトオブジェクト「ｔｈａｎｋｓ」を再生し、次いで３８６でフォームを提出する。そうでなく、ユーザが「ｎｏ」と答えた場合、あるいは認識結果の信頼度スコアが１０未満の場合は、３９０でプロンプトオブジェクト「ｒｅｔｒｙ」を実行し、その後再度プロンプトオブジェクト「ａｓｋ」を３９２で実行し、「ｒｅｃｏ＿ｄｒｉｎｋ」認識オブジェクトを３９４で起動する。

【００８１】上の例から、「ｂｉｎｄ」要素により、部分３５４、３５６、または３５８で示すような複数のメソッド起動が可能になる。所望の場合は、認識済み結果の複数の割り当ても宣言することができる。ここで説明する実施形態では、複数の割り当ておよびメソッド起動を宣言する場合、それらはドキュメントの順序で実行する。

【００８２】別の実施形態では、メソッドの引き数を渡すための規則も提供される。すなわち、一部のメソッドは引き数のリストを必要とする場合がある。これは「ａｒｇ」下位要素を使用して実現する。例えば、次のマークアップの場合、

<bind TargetElement = "OBJ" TargetMethod = "F"><arg>X</arg><arg>Y</arg></bind>

は、「ＯＢＪ．Ｆ（Ｘ，Ｙ）」に等しい。すなわち「ＯＢＪ」は、パラメータすなわち引き数「Ｘ」および「Ｙ」を用いるメソッド「Ｆ」を有するオブジェクトである。

【００８３】「ｂｉｎｄ」要素は「ｅｖｅｎｔ」属性も含むことができ、これはそのバインド要素が対象とするイベントを宣言する。例えば、マークアップ

<bind event = "onNoReco" = TargetElement = "prompt1" TargetMethod = "start"/>

は、「ｏｎＮｏＲｅｃｏ」イベントを送る際に、オブジェクト「ｐｒｏｍｐｔ１」のメソッド「ｓｔａｒｔ」を起動することを意味する。例えば図８との関連で上記で説明したように、「ｂｉｎｄ」要素を「Ｒｅｃｏ」要素の子要素として使用するのに整合するように、「ｂｉｎｄ」要素のデフォルト属性は「ｏｎＲｅｃｏ」にする。

【００８４】高レベル要素である「ｂｉｎｄ」要素は、付録の節２．４に明記するイベントをいずれも含むことができる。さらに、「ｂｉｎｄ」要素は、アクセスしてプログラムフローを指示するのに使用できる「ｓｔａｔｕｓ」属性を有する「ｏｎＥｒｒｏｒ」イベントも含むことができる。「ｂｉｎｄ」要素の他のイベントが「ｓｔａｔｕｓ」属性を有する限り、これらにもアクセスすることができる。

【００８５】認識結果の状態の確認に加えて、実行中の現在のドキュメントまたはページも確認することができる。詳細には、「ｔｅｓｔ」および「ｖａｌｕｅ」の両属性を拡張して、それを含むドキュメントのルートノードを参照する「ｈｏｓｔ」プリミティブを含ませることができる。例えば、再び図１７および１８を参照すると、ここに含まれる例は、ユーザがコーヒーを注文した際にクリームあるいは砂糖を希望するかどうかを尋ねる追加の論理を部分３５４に有する。クリームや砂糖を加え、したがって部分３５６を起動するためのフラグは、マークアップ「ｈｏｓｔ（）／ｇｅｔ＿ｄｒｉｎｋ／ｄｒｉｎｋ＝´ｃｏｆｆｅｅ´」の指定によって飲料フィールドが「コーヒー」である場合にのみオンになる。

【００８６】また、「ｂｉｎｄ」要素は音声サーバ２０４からの認識結果、値の受取り、およびそのドキュメント中への割り当てに適用できるだけでなく、メッセージ

オブジェクト（ここでは「ｓｍｅｘ」と表す。例えばクライアントデバイスで実行するアプリケーションからの）にも適用できることに留意されたい。図１７および１８の例では、クライアントデバイスで実行される電話アプリケーションが呼を検出すると、このページが実行される。部分３６０で、「ｂｉｎｄ」要素は、メッセージ「／Ｃａｌｌ＿ｃｏｎｎｅｃｔｅｄ」を受け取ると、「ｗｅｌｃｏｍｅ」プロンプトを実行または再生し、「ｒｅｃｏ＿ｄｒｉｎｋ」オブジェクトを実行することにより認識を開始する。音声サーバ２０４から受け取る認識結果と同様に、受け取るメッセージも大きく異なる可能性がある。メッセージの一部は、所望のプログラムフローを開始するために明確に規定する。受け取って処理することのできるメッセージもある（例えば、認識サーバから受け取る認識結果と同様に構文解析を行う）。例えば、これにより、キーボードから入力するテキストの自然言語パーサのようにマークアップを使用できるようになる。付録Ａのｒｅｃｏ要素は、この機能を実行するためのプロパティを含んでいる。同様に、プロンプト要素を使用し、付録Ａでさらに説明するプロパティ「ｉｎｎｅｒｔｅｘｔ」を使用することにより、動的コンテンツまたはオーディオウェーブファイル用のテキストメッセージを提供することができる。イベンティングは、認識結果のためのイベンティングと同様のものでよい。例えば、イベンティングは「ｏｎＲｅｃｅｉｖｅｄ」を含むことができるが、これは、メッセージソース（例えばクライアントデバイスで実行するアプリケーション）が、ブラウザで使用できるメッセージを有する際に送られる。

【００８７】このように、「ｓｍｅｘ」すなわちメッセージオブジェクトにより、ここに述べるようなマークアップタグを、クライアントデバイスで実行される他のコンポーネントまたはアプリケーションに拡張することが可能になる。別の例として、このメッセージオブジェクトを使用して、クライアントデバイスで実行される聴覚障害者用のＴＴＹコンポーネントと通信することができる。ＴＴＹコンポーネントは、音声認識を使用するのではなく、ユーザが入力した内容のメッセージを提供する。このメッセージはその後、認識結果を認識サーバから受け取った場合と同様に使用する。すなわち、メッセージを構文解析して、フォームのフィールドに割り当てるか、あるいは上記の「ｒｅｃｏ」、「ｇｒａｍｍａｒ」、または「ｂｉｎｄ」要素を使用して他の処理を行うことができる。このメッセージまたは「ｓｍｅｘ」オブジェクトについては、付録Ａでさらに説明する。

【００８８】「ｂｉｎｄ」要素は「ｆｏｒ」属性も含むことができ、これにより、その動作をページ上の他のオブジェクトに付することができる。例えば次のマークアップ

```
<bind for = "prompt1" event = "onComplete" targetElement = "prompt2" =targetMethod = "start"/>
```

は、オブジェクト「ｐｒｏｍｐｔ　１」がイベント「ｏｎＣｏｍｐｌｅｔｅ」を送ると、オブジェクト「ｐｒｏｍｐｔ　２」のｓｔａｒｔメソッドを起動する。

【００８９】再び図５を参照すると、ウェブサーバ２０２は、サーバ側のプラグイン宣言オーサリングツールすなわちモジュール３２０を含むことができる（例えば、マイクロソフト社によるＡＳＰまたはＡＳＰ＋、あるいはＪＳＰなど）。サーバ側のプラグインモジュール３２０は、クライアント側のマークアップと、さらにはウェブサーバ２０２にアクセスするクライアントのタイプについて固有形態のマークアップも動的に生成することができる。クライアント情報は、クライアント／サーバ関係が最初に確立されたときにウェブサーバ２０２に提供することができ、ウェブサーバ２０２は、クライアントの機能を検出するモジュールまたはルーチンを含むことができる。この方式で、サーバ側のプラグインモジュール３２０は、それぞれの音声認識シナリオ、すなわち電話機８０を通じた音声のみ、あるいはマルチモーダル型のデバイス３０に対する、クライアント側のマークアップを生成することができる。一貫性のあるクライアント側モデルを使用することにより（各アプリケーションで使用できるｒｅｃｏおよびプロンプトコントロール）、多数の異なるクライアントのアプリケーションオーサリングが大幅に容易になる。

【００９０】クライアント側マークアップの動的な生成に加えて、図８、９および１０のマークアップ例を用いた、図６に示すようなクレジットカード番号の入手などの高レベルのダイアログモジュールは、アプリケーションオーサリングで開発者が使用するために、記憶装置３２４に記憶するサーバ側コントロールとして実施することができる。一般に、高レベルダイアログモジュール３２４は、開発者が指定するパラメータに基づいて、音声のみおよびマルチモーダルの両シナリオで、クライアント側のマークアップおよびスクリプトを動的に生成する。高レベルダイアログモジュールは、開発者のニーズに適合するクライアント側のマークアップを生成するためのパラメータを含むことができる。例えば、クレジットカード情報のモジュールは、クライアント側のマークアップスクリプトが許可すべきクレジットカードの種類を指定するパラメータを含むことができる。サーバ側プラグインモジュール３２０で使用するＡＳＰ＋ページの例を図１３に示す。

【００９１】本発明について好ましい実施形態を参照して説明したが、当業者は、本発明の趣旨および範囲から逸脱せずに、形態および詳細を変更することが可能であることを理解されよう。

【００９２】付録Ａ

１　概要

以下のタグは、ドキュメントが音声を入力媒体または出

力媒体として使用することを可能にするマークアップ要素のセットである。これらのタグは、ＨＴＭＬ、ＸＨＴＭＬ、ｃＨＴＭＬ、ＳＭＩＬ、ＷＭＬなど任意のＳＧＭＬ由来のマークアップ言語に埋め込むことのできる独立型（ｓｅｌｆ－ｃｏｎｔａｉｎｅｄ）ＸＭＬになるように設計されている。本発明で使用するタグは、ワシントン州レドモンドのマイクロソフト社から入手可能な周知の方法であるＳＡＰＩ５．０に類似する。タグ、要素、イベント、属性、プロパティ、戻り値などは例示的なものに過ぎず、制限的なものと考えるべきではない。本明細書では音声およびＤＴＭＦの認識の場合の例を示すが、同様のタグは他の形の認識にも提供することができる。

【００９３】本明細書で論じる主な要素は以下である。

＜ｐｒｏｍｐｔ．．．＞　音声合成の構成およびプロンプトの再生

＜ｒｅｃｏ．．．＞　レコグナイザの構成、認識の実行、および後処理

＜ｇｒａｍｍａｒ．．．＞　入力文法リソースの指定

＜ｂｉｎｄ．．．＞　認識結果の処理

＜ｄｔｍｆ．．．＞　ＤＴＭＦの構成および制御

【００９４】２　Ｒｅｃｏ

Ｒｅｃｏ要素は、可能なユーザ入力と、入力結果の処理手段とを指定するのに使用する。したがって、その主要な要素は＜ｇｒａｍｍａｒ＞および＜ｂｉｎｄ＞にすることができ、またレコグナイザプロパティを構成するためのリソースを含む。

【００９５】Ｒｅｃｏ要素は、アップレベルブラウザではＳｔａｒｔおよびＳｔｏｐのメソッドを介してプログラム的に、またはＳＭＩＬを使用できるブラウザではＳＭＩＬコマンドを使用して起動する。この要素は、ダウンレベルブラウザ（すなわちスクリプトをサポートしないブラウザ）では、それがページ上にあることにより宣言的にアクティブであると見なす。複数の文法を並行して起動することができるように、複数のＲｅｃｏ要素を同時にアクティブと見なすことができる。

【００９６】Ｒｅｃｏは特定のモード、すなわち「自動」「シングル」または「複数」をとることもでき、これによりそれが使用可能にする認識シナリオの種類と、認識プラットフォームの振る舞いを区別する。

【００９７】２．１　Ｒｅｃｏの内容

Ｒｅｃｏ要素は、１つまたは複数の文法と、任意選択で、認識結果を調べ、関連性のある部分をそれを含むページ中の値にコピーするバインド要素のセットとを含む。

【００９８】アップレベルブラウザでは、Ｒｅｃｏは、プログラム的な起動、および個々の文法規則の非活動化をサポートする。指定しない場合は、ある認識コンテキストについて、文法のすべての最上位の規則がアクティブになることにも留意されたい。

【００９９】２．１．１　＜ｇｒａｍｍａｒ＞要素

文法要素は、インラインの、またはｓｒｃ属性を使用して参照する文法を指定するのに使用する。通例は少なくとも１つの文法（インラインまたは参照）を指定する。インライン文法はテキストベースの文法形式にすることができるのに対し、参照文法は、テキストベースまたはバイナリタイプにすることができる。複数の文法要素を指定することが可能である。複数の文法要素を指定する場合は、文法の規則を追加規則として同じ文法中に追加する。同じ名前の規則がある場合にはそれに上書きする。

【０１００】属性：

・ｓｒｃ：インライン文法を指定する場合は任意選択。含める文法のＵＲＩ。指定しない場合は、ある認識コンテキストについて、文法のすべての最上位規則がアクティブになることに留意されたい。

【０１０１】・ｌａｎｇＩＤ：任意選択。音声エンジンが使用する言語を指示するストリング。ストリングの形式は、ｘｍｌ：ｌａｎｇ定義に従う。例えば、ｌａｎｇＩＤ＝“ｅｎ－ｕｓ”は、米国英語を表す。この属性は、ｌａｎｇＩＤを文法ＵＲＩ中で指定しないときにのみ有効である。指定しない場合は、米国英語を使用する。

【０１０２】ｌａｎｇＩＤが複数の箇所で指定される場合、ｌａｎｇＩＤは、最低の有効範囲からの優先順位に従う。すなわち、リモートの文法ファイル（つまりその文法ファイル中で指定される言語ＩＤ）、次いで文法要素、次いでｒｅｃｏ要素の順となる。

```
〈grammar src="FromCity.xml" /〉
    または
〈grammar〉
    〈rule toplevel="active"〉
        〈p〉から〈/p〉
        〈ruleref name="cities" /〉
    〈/rule〉
    〈rule name="cities"〉
        〈l〉
        〈p〉 ケンブリッジ 〈/p〉
        〈p〉 シアトル 〈/p〉
        〈p〉 ロンドン 〈/p〉
        〈/l〉
    〈/rule〉
〈/grammar〉
```

ｓｒｃで参照する文法とインライン文法の両方を指定する場合は、インライン規則を参照規則に加え、同じ名前の規則があればそれに上書きする。

【０１０３】２．１．２　＜ｂｉｎｄ＞要素

バインド要素は、認識結果の値をページ中にバインドするのに使用する。

【０１０４】バインド要素によって消費される認識結果

は、認識結果を指定するためのセマンティックマークアップ言語（ＳＭＬ）を含むＸＭＬドキュメントでよい。その内容は、意味値、話された実際の単語、および信頼度スコアを含む。ＳＭＬは、代替の認識選択肢（Ｎ番目によい認識結果におけるものなど）も含むことができる。発声「Ｉ’ｄ　ｌｉｋｅ　ｔｏ　ｔｒａｖｅｌ　ｆｒｏｍ　Ｓｅａｔｔｌｅｔｏ　Ｂｏｓｔｏｎ（シアトルからボストンまで行きたい）」に対するＳＭＬドキュメントの例を下に示す。

```
<sml confidence="40">
    <travel text="シアトルからボストンまで行きたい">
        <origin_city confidence="45"> シアトル
</origin_city>
        <dest_city   confidence="35"> ボストン
</dest_city>
    </travel>
</sml>
```

【０１０５】文法中（ｉｎ－ｇｒａｍｍａｒ）認識は、セマンティックマークアップ言語すなわちＳＭＬでＸＭＬドキュメントを生成することになっているので、ＳＭＬドキュメントからバインドする値は、ＸＰａｔｈクエリを使用して参照する。また、値をバインドするページ中の要素（これはフォームコントロールである可能性が高い）は一意に識別すべきなので、これらのターゲット要素は直接参照する。

【０１０６】属性：

・ｔａｒｇｅｔＥｌｅｍｅｎｔ：必須。ＳＭＬからｖａｌｕｅの内容を割り当てる要素（Ｗ３Ｃ　ＳＭＩＬ２．０と同様）。

【０１０７】・ｔａｒｇｅｔＡｔｔｒｉｂｕｔｅ：任意選択。ＳＭＬからｖａｌｕｅの内容を割り当てるターゲット要素の属性（ＳＭＩＬ２．０のａｔｔｒｉｂｕｔｅＮａｍｅ属性と同様）。指定しない場合は、「ｖａｌｕｅ」になる。

【０１０８】・ｔｅｓｔ：任意選択。認識結果を割り当てる際の条件を指示するＸＭＬ　Ｐａｔｔｅｒｎ（Ｗ３Ｃ　ＸＭＬＤＯＭ仕様と同様）ストリング。デフォルト条件は真。

【０１０９】・ｖａｌｕｅ：必須。ターゲット要素に割り当てる認識結果ドキュメントの値を指定するＸＰＡＴＨ（Ｗ３Ｃ　ＸＭＬ　ＤＯＭ仕様と同様）ストリング。

【０１１０】例：上記のＳＭＬのリターンを与えられると、以下のｒｅｃｏ要素はバインドを使用して、ｏｒｉｇｉｎ＿ｃｉｔｙおよびｄｅｓｔ＿ｃｉｔｙ中の値を、ターゲットページの要素ｔｘｔＢｏｘＯｒｉｇｉｎおよびｔｘｔＢｏｘＤｅｓｔに転送する。

```
〈input name:"txtBoxOrigin" type="text"/〉
〈input name:"txtBoxDest" type="text"/〉

〈reco id="travel"〉
    〈grammar src="./city.xml" /〉

    〈bind      targetElement="txtBoxOrigin"
          value="//origin_city" /〉
    〈bind      targetElement="txtBoxDest"
          value="//dest_city" /〉
〈/reco〉
```

このバインドは、バインド操作の事前条件としてｄｅｓｔ＿ｃｉｔｙ結果の信頼度属性にテストを行う以下の例のように条件付きの場合もある。

```
〈bind targetElement="txtBoxDest"
    value:"//dest_city"

    test="/sml/dest_city[@confidence $gt$ 40]"
/〉
```

バインド要素は、ダウンレベルまたはアップレベルのブラウザで認識結果を処理する単純な宣言的手段である。より複雑な処理の場合、アップレベルブラウザによってサポートされるｒｅｃｏＤＯＭオブジェクトは、ｏｎＲｅｃｏイベントハンドラを実装して、プログラム的なスクリプト分析と認識の戻しの後処理を行えるようにする。

【０１１１】２．２　属性およびプロパティ

以下の属性はすべてのブラウザでサポートされ、プロパティはアップレベルブラウザによってサポートされる。

【０１１２】２．２．１　属性

以下のＲｅｃｏの属性は、ダイアログターンのために音声レコグナイザを構成するのに使用する。

【０１１３】・ｉｎｉｔｉａｌＴｉｍｅｏｕｔ：任意選択。認識の開始から音声の検出までのミリ秒単位の時間。この値は認識プラットフォームに渡され、これを超えた場合は、ｏｎＳｉｌｅｎｃｅイベントが認識プラットフォームから提供される（２．４．２参照）。指定し

ない場合、音声プラットフォームはデフォルト値を使用する。

【０１１４】・ｂａｂｂｌｅＴｉｍｅｏｕｔ：任意選択。音声の検出後にレコグナイザが結果を戻さなければならないミリ秒単位の期間。自動モードおよびシングルモードのｒｅｃｏの場合、これは音声検出からｓｔｏｐ呼び出しまでの期間に該当する。「複数」モードのｒｅｃｏの場合、このタイムアウトは、音声検出から各認識の戻しまでの期間に相当する。すなわち、各結果の戻しまたは他のイベントの後にこの期間を再び開始する。このタイムアウトを超えると、エラーの発生の有無に応じて異なるイベントを投入する。例えば、発声が例外的に長い場合など、レコグナイザがなおオーディオを処理している場合は、ステータスコード１３により（２．４．４参照）ｏｎＮｏＲｅｃｏイベントを投入する。ただし、何らかの他の理由でこのタイムアウトを超えた場合はレコグナイザのエラーである可能性がより高くなり、ｏｎＴｉｍｅｏｕｔイベントを投入する。指定しない場合、音声プラットフォームは内部値を使用する。

【０１１５】・ｍａｘＴｉｍｅｏｕｔ：任意選択。認識の開始からブラウザに結果を戻すまでのミリ秒単位の期間。これを超えると、ブラウザによってｏｎＴｉｍｅｏｕｔイベントが投入され、これにより分散環境におけるネットワークまたはレコグナイザの障害に対処（ｃａｔｅｒ　ｆｏｒ）する。「複数」モードのｒｅｃｏの場合は、ｂａｂｂｌｅＴｉｍｅｏｕｔと同様に、各認識の戻しまたは他のイベントの後にこの期間を再度開始する。ｍａｘＴｉｍｅｏｕｔ属性は、ｉｎｉｔｉａｌＴｉｍｅｏｕｔとｂａｂｂｌｅＴｉｍｅｏｕｔの合計よりも大きくするか、または等しくすべきであることに留意されたい。指定しない場合、この値はブラウザのデフォルトになる。

【０１１６】・ｅｎｄＳｉｌｅｎｃｅ：任意選択。自動モードのＲｅｃｏの場合、認識結果を戻すまでの、音声があってはならない発話終了後のミリ秒単位の無音期間。自動モード以外のモードのｒｅｃｏについては無視する。指定しない場合は、プラットフォームの内部値になる。

【０１１７】・ｒｅｊｅｃｔ：任意選択。認識拒絶の閾値。これを下回ると、プラットフォームは「ｎｏ　ｒｅｃｏ」イベントを投入する。指定しない場合、音声プラットフォームはデフォルト値を使用する。信頼度スコアは、０から１００の範囲（整数）。拒絶値はこの範囲内にある。

【０１１８】・ｓｅｒｖｅｒ：任意選択。音声プラットフォームのＵＲＩ（タグインタープリタと認識プラットフォームをまとめて配置しない場合に使用する）。値の例は、ｓｅｒｖｅｒ＝ｐｒｏｔｏｃｏｌ：／／ｙｏｕｒｓｐｅｅｃｈｐｌａｔｆｏｒｍなどとなる。アプリケーションの作成者（ａｕｔｈｏｒ）は、ＵＲＩストリングにクエリストリングを加えることにより、音声プラットフォームに固有の設定を提供することもできる。例：ｐｒｏｔｏｃｏｌ：／／ｙｏｕｒｓｐｅｅｃｈｐｌａｔｆｏｒｍ？ｂａｒｇｅｉｎＥｎｅｒｇｙＴｈｒｅｓｈｏｌｄ＝０．５。

【０１１９】・ｌａｎｇＩＤ：任意選択。音声エンジンが使用する言語を指定するストリング。ストリング形式は、ｘｍｌ：ｌａｎｇ　定義に従う。例えば、ｌａｎｇ＝“ｅｎ－ｕｓ”は米国英語を表す。この属性は、文法要素中でｌａｎｇＩＤを指定しない場合のみに有効である（２．１．１参照）

・ｍｏｄｅ：任意選択。とるべき認識モードを指定するストリング。指定しない場合は、「自動」モードになる。

【０１２０】２．２．２　プロパティ

以下のプロパティは、認識プロセスによって戻される結果を含む（これらはアップレベルブラウザにサポートされる）。

【０１２１】・ｒｅｃｏＲｅｓｕｌｔ：読み取り専用。認識の結果、２．１．２で述べたように、セマンティックマークアップ言語（ＳＭＬ）を含むＸＭＬ　ＤＯＭノードオブジェクト中に保持される。認識が行われなかった場合、このプロパティはヌルに戻る。

【０１２２】・ｔｅｘｔ：読み取り／書き込み。認識された単語のテキストを保持するストリング（すなわち、読み取りモードにおけるｒｅｃｏＲｅｓｕｌｔ中のＳＭＬ認識の戻しの中の最上位要素のテキスト属性の内容を表す省略表現）。書き込みモードでは、ストリングを割り当てることができ、次いでそのストリングが認識結果に対応するものとしてそれを構文解析する。書き込みモードでは、このマークアップタグおよびその処理を、クライアントデバイスの他のコンポーネントまたはアプリケーションに拡張することができる。このストリングは、「ｓｍｅｘ」メッセージオブジェクトから得られる。

【０１２３】・ｓｔａｔｕｓ：読み取り専用。認識プラットフォームが返すステータスコード。可能な値は、認識が成功した場合の０、あるいは障害値－１から－４（Ｓｔａｒｔメソッド（節２．３．１）およびＡｃｔｉｖａｔｅメソッド（節２．３．４）で可能な例外で定義する）、およびレコグナイザイベントを受け取った際にセットされるステータス－１１から－１５（２．４参照）。

【０１２４】２．３　オブジェクトメソッド

ｒｅｃｏの起動および文法の起動は、ＲｅｃｏのＤＯＭオブジェクト中の以下のメソッドを使用して制御することができる。これらのメソッドにより、アップレベルブラウザはＲｅｃｏオブジェクトの開始および中止、進行中の認識のキャンセル、個々の文法のトップレベルの規則の起動および非活動化を行うことができる（アップレ

ベルブラウザのみ）。

【０１２５】２．３．１　Ｓｔａｒｔ

Ｓｔａｒｔメソッドは、明示的には非活動化していない認識コンテキストについてのすべての最上位規則をアクティブな文法として使用して認識プロセスを開始する。

【０１２６】構文：Ｏｂｊｅｃｔ．Ｓｔａｒｔ（　）

戻り値：なし

例外：このメソッドは、非ゼロのステータスコードをセットし、障害があった際はｏｎＮｏＲｅｃｏイベントを発生させる。可能性のある障害には、文法が存在しない（ｒｅｃｏステータス＝－１）、文法のコンパイルの失敗、存在しないＵＲＩなど様々な原因になりうる文法のロードの失敗（ｒｅｃｏステータス＝－２）、あるいは音声プラットフォームのエラー（ｒｅｃｏステータス＝－３）などが含まれる。

【０１２７】２．３．２　Ｓｔｏｐ

Ｓｔｏｐメソッドは、認識プロセスを終了する呼び出しである。Ｒｅｃｏオブジェクトはオーディオの記録を中止し、レコグナイザは、記録が中止される時点までに受け取ったオーディオについての認識結果を戻す。Ｒｅｃｏが使用するすべての認識リソースは解放され、その文法は非活動化される。（このメソッドは、自動モードによる通常の認識には明示的に使用する必要がないことに留意されたい。これは、レコグナイザ自体が、完全な文を認識した後のエンドポイント検出においてｒｅｃｏオブジェクトを中止するからである。）Ｒｅｃｏが開始されていない場合、この呼び出しは効果を持たない。

【０１２８】構文：Ｏｂｊｅｃｔ．Ｓｔｏｐ（　）

戻り値：なし

例外：なし

【０１２９】２．３．３　Ｃａｎｃｅｌ

Ｃａｎｃｅｌメソッドは、レコグナイザへのオーディオの供給を中止し、文法を非活動化し、レコグナイザを解放し、すべての認識結果を破棄する。ブラウザは、キャンセルされた認識についての認識結果は破棄する。レコグナイザが開始されていない場合、この呼び出しは効果を持たない。

【０１３０】構文：Ｏｂｊｅｃｔ．Ｃａｎｃｅｌ（　）

戻り値：なし

例外：なし

【０１３１】２．３．４　Ａｃｔｉｖａｔｅ

Ａｃｔｉｖａｔｅメソッドは、文脈自由文法（ＣＦＧ）の最上位規則を起動する。起動は、「開始された」認識プロセス中には効果を持たないので、認識が開始する前に呼び出さなければならない。明示的に非活動化していない認識コンテキストについてのすべての文法の最上位規則は、すでにアクティブであると見なすことに留意されたい。

【０１３２】構文：Ｏｂｊｅｃｔ．Ａｃｔｉｖａｔｅ（ｓｔｒＮａｍｅ）

パラメータ：

・ｓｔｒＮａｍｅ：必須。起動する規則名。

戻り値：なし

例外：なし

【０１３３】２．３．５　Ｄｅａｃｔｉｖａｔｅ

このメソッドは、文法中のトップレベル規則を非活動化する。その規則が存在しない場合、このメソッドは効果を持たない。

構文：Ｏｂｊｅｃｔ．Ｄｅａｃｔｉｖａｔｅ（ｓｔｒＮａｍｅ）

パラメータ：

・ｓｔｒＮａｍｅ：必須。非活動化する規則名。空ストリングはすべての規則を非活動化する。

戻り値：なし

例外：なし

【０１３４】２．４　Ｒｅｃｏイベント

Ｒｅｃｏ　ＤＯＭオブジェクトは以下のイベントをサポートし、そのハンドラはｒｅｃｏ要素の属性として指定することができる。

【０１３５】２．４．１　ｏｎＲｅｃｏ：このイベントは、レコグナイザが、そのブラウザで利用することのできる認識結果を得ると起動される。自動モードのｒｅｃｏの場合、このイベントは認識プロセスを自動的に中止し、リソースをクリアする（２．３．２参照）。ｏｎＲｅｃｏは通例、認識結果のプログラム的な分析と、ページ中への結果の処理に使用される。

【０１３６】構文：

【０１３７】

【表１】

インラインHTML	<Reco onReco ="handler">
イベントプロパティ	Object.onReco = handler; Object.onReco = GetRef("handler");

【０１３８】イベントオブジェクト情報：

【表２】

【０１３９】

バブル	なし
起動するには	ユーザが何かを言う
デフォルトアクション	認識結果オブジェクトを戻す

【０１４０】イベントプロパティ：イベントハンドラはプロパティを直接受け取ることはないが、ハンドラはデータについてこのイベントオブジェクトに照会を行うことができる（下記の例のイベントオブジェクトの使用を参照のこと）。

【０１４１】例

次のＸＨＴＭＬの断片ではｏｎＲｅｃｏを使用して、認識結果を構文解析し、その値を適切なフィールドに割り当てるスクリプトを呼び出している。

```
    <input name:"txtBoxOrigin" type="text" />
    <input name:"txtBoxDest" type="text" />
    <reco onReco="processCityRecognition()"/>
        <grammar src="/grammars/cities.xml" />
    </reco>

    <script><![CDATA[

        function processCityRecognition () {
            smlResult =
event.srcElement.recoResult;

            origNode =
smlResult.selectSingleNode("//origin_city");
            if (origNode != null)
txtBoxOrigin.value = origNode.text;

            destNode =
smlResult.selectSingleNode("//dest_city");
            if (destNode != null) txtBoxDest.value
= destNode.text;
        }
    ]]> </script>
```

【０１４２】２．４．２　ｏｎＳｉｌｅｎｃｅ：ｏｎＳ

インラインHTML	<Reco onSilence="handler" ...>
イベントプロパティ (ECMAScript)	Object.onSilence = handler Object.onSilence = GetRef("handler");

【０１４５】イベントオブジェクト情報：　　【表４】

【０１４６】

バブル	なし
起動するには	initialTimeout 属性で指定される期間中にレコグナイザが音声を検出しなかった
デフォルトアクション	ステータス＝-11 にセット

【０１４７】イベントプロパティ：イベントハンドラ　　は、プロパティを直接受け取ることはないが、ハンドラはデータについてこのイベントオブジェクトに照会を行うことができる（下記の例のイベントオブジェクトの使用を参照のこと）。

【０１４８】例

次のＸＨＴＭＬの断片ではｏｎＲｅｃｏを使用して、認識結果を構文解析し、その値を適切なフィールドに割り当てるスクリプトを呼び出している。

```
    <input name:"txtBoxOrigin" type="text" />
    <input name:"txtBoxDest" type="text" />
    <reco onReco="processCityRecognition()"/>
        <grammar src="/grammars/cities.xml" />
    </reco>

    <script><![CDATA[

        function processCityRecognition () {
            smlResult =
event.srcElement.recoResult;

            origNode =
smlResult.selectSingleNode("//origin_city");
            if (origNode != null)
txtBoxOrigin.value = origNode.text;

            destNode =
smlResult.selectSingleNode("//dest_city");
            if (destNode != null) txtBoxDest.value
= destNode.text;
        }
    ]]> </script>
```

【０１４９】２．４．３　ｏｎＳｉｌｅｎｃｅ：ｏｎＳｉｌｅｎｃｅは、ＲｅｃｏのｉｎｉｔｉａｌＴｉｍｅｏｕｔ属性で指定された時間が過ぎる前に、認識プラットフォームが検出した無音声のイベントに対処する（２．２．１参照）。このイベントは、自動認識モードの認識プロセスを自動的にキャンセルする。

【０１５０】構文：

【０１５１】

【表３】

はデータについてイベントオブジェクトに照会を行うことができる。

【０１４８】２．４．３　ｏｎＴｉｍｅｏｕｔ

ｏｎＴｉｍｅｏｕｔは、通例は音声プラットフォームからのエラーを反映する２タイプのイベントを扱う。

【０１４９】・認識が完了する前にｍａｘＴｉｍｅ属性で指定された期間を過ぎた（２．２．１参照）ことを通知する、タグインタープリタが投入するイベントを扱う。このイベントは通例、分散型アーキテクチャで生じうる問題を反映する。

【０１５０】・また、（ｉｉ）認識が開始されたが、ｂａｂｂｌｅＴｉｍｅｏｕｔで指定された期間内に認識がないまま処理が中止した際に、音声認識プラットフォームが投入するイベントも扱う（２．２．１参照）。

【０１５１】このイベントは、認識プロセスを自動的にキャンセルする。

【０１５２】構文：

【０１５３】

【表５】

インラインHTML	<Reco onTimeout ="handler" ...>
イベントプロパティ (ECMAScript)	Object.onTimeout = handler Object.onTimeout = GetRef("handler");

【０１５４】イベントオブジェクト情報：

【０１５５】

【表６】

バブル	なし
起動するには	認識の中止前に、maxTime 属性で設定された期間が過ぎるとブラウザが投入する
デフォルトアクション	reco ステータスを-12 にセットする

【０１５６】イベントプロパティ：イベントハンドラはプロパティを直接受け取ることはないが、ハンドラはデータについてこのイベントオブジェクトに照会を行うことができる。

【０１５７】２．４．４　ｏｎＮｏＲｅｃｏ：ｏｎＮｏＲｅｃｏは、有効な認識結果を戻すことができない際に音声認識プラットフォームが投入するイベント用のハンドラである。それが発生しうる異なるケースは、ステータスコードで区別する。このイベントは認識プロセスを自動的に中止する。

【０１５８】構文：

【０１５９】

【表７】

インラインHTML	<Reco onNoReco ="handler" >
イベントプロパティ	Object.onNoReco = handler; Object.onNoReco = GetRef("handler");

【０１６０】イベントオブジェクト情報：

【０１６１】

【表８】

バブル	なし
起動するには	レコグナイザが音を検出するが、発声を解釈することができない
デフォルトアクション	ステータスプロパティをセットし、ヌルの認識結果を戻す。ステータスコードは以下のようにセットする。 ステータス－13：音が検出されたが、解釈できる音声がなかった場合 ステータス－14：いくらかの音声が検出され解釈されたが、信頼度が不十分であるために拒絶された場合（閾値の設定については2.2.1 の拒絶の属性を参照のこと） ステータス－15：音声が検出され解釈されたが、音声の検出からbabbleTimeout 属性で指定された期間までに完全な認識を戻すことができなかった場合（2.2.1 参照）

【０１６２】イベントプロパティ：イベントハンドラはプロパティを直接受け取ることはないが、データについてこのイベントオブジェクトに照会を行うことができる。

【０１６３】３　プロンプト

プロンプト要素は、システム出力を指定するのに使用する。その内容は以下の１つまたは複数にすることができる。

【０１６４】・インラインテキストまたは参照テキスト。これは、韻律的な（ｐｒｏｓｏｄｉｃ）またはその他の音声出力情報でマークアップすることができる。

・レンダリング時にそれを含むドキュメントから取り出す変数値。

・オーディオファイルへのリンク。

【０１６５】プロンプト要素は、ダウンレベルブラウザによって宣言的に解釈する（あるいはＳＭＩＬコマンドで起動する）ことも、アップレベルブラウザのオブジェクトメソッドによって宣言的に解釈することもできる。

【０１６６】３．１　プロンプト内容

プロンプト要素は、テキストまたはオーディオファイルへのリファレンスの形で、あるいはこの両方の形でシステム出力用のリソースを含む。

【０１６７】簡単なプロンプトは、出力に必要なテキストだけを指定すればよい。例えば、

```
<prompt id="Welcome">
    ＡＣＭＥ天気予報へのお電話ありがとうございます
</prompt>
```

この簡単なテキストは、下記に説明する種類のどのマークアップもさらに含むことができる。

【０１６８】３．１．１　音声合成マークアップ

このプロンプト要素の内部では、どの形式の音声合成マークアップ言語でも使用することができる。（この形式は、３．２．１で説明する「ｔｔｓ」属性で指定することができる。）次の例は、その中の特定の単語を強調する命令を含むテキストを示している。

```
<prompt id="giveBalance">
    あなたの口座の残高は＜ｅｍｐｈ＞５ドル＜／ｅｍｐｈ＞です
</prompt>
```

【０１６９】３．１．２　動的な内容

このプロンプトの実際の内容は、プロンプトの出力の直前にクライアントで計算する必要がある場合がある。例えば特定の値を確定するには、ある変数にその値をデリファレンスする必要がある。この値要素はこの目的に使用することができる。

【０１７０】値要素

ｖａｌｕｅ：任意選択。ドキュメント中の要素の値を取り出す。

属性：

・ｔａｒｇｅｔＥｌｅｍｅｎｔ：任意選択。ｈｒｅｆまたはｔａｒｇｅｔＥｌｅｍｅｎｔを指定しなければならない。取り出す値を含む要素のＩＤ。

・ｔａｒｇｅｔＡｔｔｒｉｂｕｔｅ：任意選択。値を取り出す要素の属性。

・ｈｒｅｆ：任意選択。オーディオセグメントのＵＲＩ。両方ある場合には、ｈｒｅｆがｔａｒｇｅｔＥｌｅｍｅｎｔを上書きする。

【０１７１】ｔａｒｇｅｔＥｌｅｍｅｎｔ属性は、それを含むドキュメント中の要素を参照するのに使用される。ｔａｒｇｅｔＥｌｅｍｅｎｔによってＩＤが指定された要素の内容を、合成するテキストに挿入する。所望の内容がその要素の属性に保持されている場合、ｔａｒｇｅｔＡｔｔｒｉｂｕｔｅを使用して、ｔａｒｇｅｔＥ

ｌｅｍｅｎｔの必要な属性を指定することができる。これは、例えば、ＨＴＭＬフォームコントロール中の値をデリファレンスするのに有用である。下の例では、「ｔｘｔＢｏｘＯｒｉｇｉｎ」要素および「ｔｘｔＢｏｘＤｅｓｔ」要素の「ｖａｌｕｅ」属性を、プロンプトの出力前にテキストに挿入している。

```
〈prompt id="Confirm"〉
    あなたが行きたいのは
    〈value targetElement:"txtBoxOrigin"
targetAttribute:"value" /〉

    から
    〈value targetElement:"txtBoxDest"
targetAttribute:"value" /〉
    ですか？
〈/prompt〉
```

【０１７２】３．１．３　オーディオファイル

```
<prompt>
    ピーッという音がしたらメッセージを録音してください
    <value href="/wav/beep.wav" />
</prompt>
```

【０１７３】３．１．４　参照プロンプト

インラインの内容を指定する代わりに、ｓｒｃ属性を空要素とともに使用し、ＵＲＩを介して外部の内容を参照することができる。例えば、

```
〈prompt id="Welcome"
src="/ACMEWeatherPrompts#Welcome" /〉
```

ｓｒｃ属性の対象は、インラインプロンプトに指定する上記の内容の任意部分またはすべてを保持することができる。

【０１７４】３．２　属性およびプロパティ

このプロンプト要素は、以下の属性（ダウンレベルブラウザ）およびプロパティ（ダウンレベルおよびアップレベルブラウザ）を保持する。

【０１７５】３．２．１　属性

・ｔｔｓ：任意選択。テキストから音声への合成用のマークアップ言語タイプ。デフォルトは「ＳＡＰＩ５」。

【０１７６】・ｓｒｃ：インラインプロンプトを指定する場合は任意選択。参照するプロンプトのＵＲＩ（３．１．４参照）。

【０１７７】・ｂａｒｇｅｉｎ：任意選択。整数。プロンプトの開始から、人間の聴者が再生を中断できるようになるまでのミリ秒単位の時間。デフォルトは無限、すなわちバージインを許可しない。ｂａｒｇｅｉｎ＝０にすると、即時のバージインが可能になる。これは、プラットフォームがサポートするどの種のバージインにも該当する。ｒｅｃｏを開始する時間にどちらを使用可能にするかに応じて、キーワードまたはエネルギーベースのバージイン時間をこの方式で構成することができる。この値要素は、合成したプロンプトの代わりに、あるいはその中で再生するあらかじめ記録したオーディオファイルを参照するのにも使用することができる。次の例では、プロンプトの最後にビープ音を鳴らしている。

【０１７８】・ｐｒｅｆｅｔｃｈ：任意選択。ページをロードする際にプロンプトを直ちに合成して、ブラウザにキャッシュするかどうかを示すブールフラグ。デフォルトは偽。

【０１７９】３．２．２　プロパティ

アップレベルブラウザは、プロンプトのＤＯＭオブジェクト中の以下のプロパティをサポートする。

【０１８０】・ｂｏｏｋｍａｒｋ：読み取り専用。遭遇した最後の合成ブックマークのテキストを記録するストリングオブジェクト。

【０１８１】・ｓｔａｔｕｓ：読み取り専用。音声プラットフォームから戻されるステータスコード。

【０１８２】・ｉｎｎｅｒｔｅｘｔ：読み取り専用。このプロパティはプロンプトのテキストの複写（ｔｒａｎｓｃｒｉｐｔｉｏｎ）を提供し、それがシンセサイザに送られる。例えば、あるプロンプトがオーディオウェーブファイルの再生を含む場合、このプロパティはそのプロンプトのテキストバージョン（オーディオウェーブファイルとともに記憶することが多い）を提供し、これはその後、例えばクライアントデバイスで実行するコンポーネントまたはアプリケーションにプロンプトのテキストバージョンを提供することにより、表示するか、またはその他の形で使用することができる。またｉｎｎｅｒｔｅｘｔプロパティを使用して、動的コンテンツを含むプロンプトのテキストバージョンも提供することができる。

【０１８３】３．３　プロンプトメソッド

プロンプトの再生は、プロンプトのＤＯＭオブジェクト中の以下のメソッドを使用して制御することができる。

この方式により、アップレベルブラウザは、プロンプトオブジェクトを開始および停止し、進行中のプロンプトを一時停止および再開し、合成音声のスピードおよび音量を変えることができる。

【０１８４】３．３．１　Ｓｔａｒｔ

プロンプトの再生を開始する。引き数が与えられない限り、このメソッドはオブジェクトの内容を再生する。所与の時間に単一のプロンプトオブジェクトだけが「開始される」と考えられるので、Ｓｔａｒｔを連続して呼び出すとすべての再生が連続的に再生される。

【０１８５】構文：Ｏｂｊｅｃｔ．Ｓｔａｒｔ（［ｓｔｒＴｅｘｔ］）

パラメータ：

・ｓｔｒＴｅｘｔ：シンセサイザに送信するテキスト。存在する場合にはこの引き数がオブジェクトの内容を上書きする。

戻り値：なし

例外：サーバがすでにオーディオバッファを開放している場合には、ステータス＝－１にセットし、ｏｎＣｏｍｐｌｅｔｅイベントを発生させる。

【０１８６】３．３．２　Ｐａｕｓｅ

オーディオバッファをフラッシュすることなく再生を一時停止する。このメソッドは、再生を一時停止または停止している場合には効果を持たない。

構文：Ｏｂｊｅｃｔ．Ｐａｕｓｅ（　）；

戻り値：なし

例外：なし

【０１８７】３．３．３　Ｒｅｓｕｍｅ

オーディオバッファをフラッシュすることなく再生を再開する。このメソッドは、再生が一時停止状態にない場合は効果を持たない。

構文：Ｏｂｊｅｃｔ．Ｒｅｓｕｍｅ（　）；

戻り値：なし

例外：再開が失敗した際に例外を投入する。

【０１８８】３．３．４　Ｓｔｏｐ

再生がまだ中止されていない場合に再生を中止し、オーディオバッファをフラッシュする。再生がすでに中止されている場合、このメソッドは単にオーディオバッファをフラッシュする。

構文：Ｏｂｊｅｃｔ．Ｓｔｏｐ（　）；

戻り値：なし

例外：なし

【０１８９】３．３．５　Ｃｈａｎｇｅ

再生の速度および／または音量を変更する。Ｃｈａｎｇｅは再生中に呼び出すことができる。

【０１９０】構文：Ｏｂｊｅｃｔ．Ｃｈａｎｇｅ（ｓｐｅｅｄ，ｖｏｌｕｍｅ）；

パラメータ：

・ｓｐｅｅｄ：必須。変化させる係数。

ｓｐｅｅｄ＝２．０は、現在の速度を２倍にすることを意味し

ｓｐｅｅｄ＝０．５は、現在の速度の２分の１にすることを意味し、

ｓｐｅｅｄ＝０は、デフォルト値に戻すことを意味する。

・ｖｏｌｕｍｅ：必須。変化させる係数。

ｖｏｌｕｍｅ＝２．０は、現在の音量を倍にすることを意味し、

ｖｏｌｕｍｅ＝０．５は、現在の音量を半分にすることを意味し、

ｖｏｌｕｍｅ＝０は、デフォルト値に戻すことを意味する。

戻り値：なし

例外：なし

【０１９１】３．３．６　プロンプトコントロールの例

次の例は、キーワードバージインの機構をサポートしないプラットフォームに対して、上記のメソッドを使用するプロンプトコントロールをオーサリングする仕組みを示している。

```
<html>
<title>プロンプトコントロール</title>
<head>
  <script>
    <!--
        function checkKWBargein() {
            news.change (1.0, 0.5); // 確認中は音量を下げる
            if (keyword.text == "") { // 結果が閾値以下である場合
                news.change (1.0, 2.0); // 音量を元に戻す
                keyword.Start (); // 認識を再開
            } else {
                news.Stop (); // キーワード検出！プロンプトを中止
                // 必要事項を行う
            }
        }
```

```
//
  </script>
<script for="window" event="onload">
   <!--
            news.Start (); // keyword.Start ();
   //
 </script>
   </head>
<body>
  <prompt id="news" bargein="0">
```

水曜日の株式市場も、投資家が、来週の連邦準備理事会の会合に先立ち大きな動きにつながる材料を得られなかったことから展開に活気がありませんでした。ハイテク銘柄中心のナスダック総合指数は４２．５１ポイント下落し、２１５６．２６で取引を終えました。ダウジョーンズ工業平均株価は、午後に入って反騰がなく１７．０５ポイント下落して１０８６６．４６で取引を終えました。

```
・〈!—
  〈/prompt〉
    〈reco      id="keyword"〉
        reject="70"
        onReco="checkKWBargein()" 〉
        〈grammar
    src=http://denali/news bargein grammar.xml  /〉
      〈/reco〉
  〈/body〉
  〈/html〉
```

【０１９２】３．４　プロンプトイベント

インラインHTML	<prompt onBookmark ="handler"...>
イベントプロパティ	Object.onBookmark = handler Object.onBookmark = GetRef("handler");

【０１９６】イベントオブジェクト情報：　　【表１０】
【０１９７】

バブル	なし
起動するには	レンダリングしたストリング中のブックマークに遭遇する
デフォルトアクション	ブックマークストリングを戻す

【０１９８】イベントプロパティ：イベントハンドラはプロパティを直接受け取ることはないが、ハンドラはデータについてこのイベントオブジェクトに照会を行うことができる。

【０１９９】３．４．２　ｏｎＢａｒｇｅｉｎ：ユーザのバージインイベントを検出すると発生する。（例えばエネルギー検出やキーワード認識など、何がバージインイベントを構成するかの決定はプラットフォームによることに留意されたい。）このイベントハンドラを指定しても、自動的にバージイン機能がオンになるわけではない。

【０２００】構文：
【０２０１】
【表１１】

プロンプトＤＯＭオブジェクトは以下のイベントをサポートするが、そのハンドラはプロンプト要素の属性として指定することができる。

【０１９３】３．４．１　ｏｎＢｏｏｋｍａｒｋ
合成ブックマークに遭遇すると発生する。このイベントは再生を一時停止しない。

【０１９４】構文：
【０１９５】
【表９】

インラインHTML	<prompt onBargein ="handler" ...>
イベントプロパティ	Object.onBargein = handler Object.onBargein = GetRef("handler");

【０２０２】イベントオブジェクト情報：　　【表１２】
【０２０３】

バブル	なし
起動するには	バージインイベントに遭遇する
デフォルトアクション	なし

【０２０４】イベントプロパティ：イベントハンドラはプロパティを直接受け取ることはないが、ハンドラはデータについてこのイベントオブジェクトに照会を行うことができる。

【０２０５】３．４．３　ｏｎＣｏｍｐｌｅｔｅ：プロンプトの再生が最後に達するか、または例外（上記に定義）に遭遇すると発生する。

【０２０６】構文：
【０２０７】　　【表１３】

インラインHTML	<prompt onComplete ="handler" ...>
イベントプロパティ	Object. onComplete = handler Object. onComplete = GetRef("handler");

【０２０８】イベントオブジェクト情報：　　【表１４】
【０２０９】

バブル	なし
起動するには	プロンプト再生が完了する
デフォルトアクション	再生が正常に完了した場合はステータス＝0 にセットし、その他の場合は上記に指定するようにステータスをセットする

【０２１０】イベントプロパティ：イベントハンドラはプロパティを直接受け取ることはないが、ハンドラはデータについてこのイベントオブジェクトに照会を行うことができる。

【０２１１】３．４．４　ブックマークおよびイベントの使用

次の例は、プロンプトの出力中にバージインが行われた場合に、ブックマークイベントを使用して、出発地の訂正か目的地の提供のいずれかであるユーザ応答の意味を判定する仕組みを示している。ｏｎＢａｒｇｅｉｎハンドラが、プロンプト中に遭遇した最後のブックマークにグローバルな「ｍａｒｋ」変数を設定するスクリプトを呼び出し、この「ｍａｒｋ」の値をｒｅｃｏの後処理関数（「ｈｅａｒｄ」）で使用して、正しい値をセットしている。

【０２１２】

```
<script><![CDATA[
    var mark;
    function interrupt( ) {
        mark = event.srcElement.bookmark;
    }
    function ProcessCityConfirm() {
        confirm.stop(); // オーディオバッファをフラッシュする
        if (mark == "mark_origin_city")
            txtBoxOrigin.value =
event.srcElement.text;
        else
            txtBoxDest.value =
event.srcElement.text;
    }
```

```
    ]]></script>
    <body>
    <input name="txtBoxOrigin" value="Seattle"
type="text"/>
    <input name="txtBoxDest" type="text" />
    ...
    <prompt id="confirm" onBargein="interrupt()"
bargein="0">
        <bookmark mark="mark_origin_city" />
        <value targetElement="origin"
targetAttribute="value" />から
        <bookmark mark="mark_dest_city"
/>行きたい行先地を言って下さい
    </prompt>
    <reco onReco="ProcessCityConfirm()" >
        <grammar src="/grm/1033/cities.xml" />
    </reco>
    ...
    </body>
```

【０２１３】４　ＤＴＭＦ

ＤＴＭＦ認識オブジェクトを作成する。このオブジェクトは、インラインのマークアップ言語構文を使用して、あるいははスクリプト中にインスタンス化することができる。起動すると、ＤＴＭＦにより、プロンプトオブジェクトがバージインイベントを発生することができる。下記でＤＴＭＦとの関連で説明するタグおよびイベンティング、および節５で説明する呼制御は、一般には、音声ブラウザ２１６とメディアサーバ２１４間の対話に関連するものであることに留意されたい。

【０２１４】４．１　内容

・ｄｔｍｆｇｒａｍｍａｒ：インライン文法

・ｂｉｎｄ：ＤＴＭＦの変換結果を適切なフィールドに割り当てる

属性：

・ｔａｒｇｅｔＥｌｅｍｅｎｔ：必須。部分的な認識結果を割り当てる要素（参照：Ｗ３Ｃ　ＳＭＩＬ２．０に同じ）。

・ｔａｒｇｅｔＡｔｔｒｉｂｕｔｅ：認識結果を割り当てるターゲット要素の属性（参照：ＳＭＩＬ２．０に同じ）。デフォルトは「ｖａｌｕｅ」。

・ｔｅｓｔ：割り当ての条件。デフォルトは真。

【０２１５】

例１：テキストにキーをマッピングする

```
〈input type="text" name:"city"/〉
〈DTMF id:"city_choice" timeout="2000"
numDigits="1"〉
        〈dtmfgrammar〉
            〈key value="1"〉シアトル〈/key〉
            〈key value="2"〉ボストン〈/key〉
        〈/dtmfgrammar〉

        〈bind targetElement="city"
targetAttribute:"value" /〉
    〈/DTMF〉
```

「ｃｉｔｙ＿ｃｈｏｉｃｅ」を起動して、ユーザが１を押すと「Ｓｅａｔｔｌｅ」が入力フィールドに割り当てられ、２を押すと「Ｂｏｓｔｏｎ」が割り当てられ、その他の場合は何も割り当てられない。

【０２１６】例２：どのようにしてＤＴＭＦを複数フィールドに使用することができるか

```
〈input type="text" name:"area_code"/〉
〈input type="text" name:"phone_number" /〉
〈DTMF id:"areacode" numDigits="3"
onReco="extension.Activate()"〉
        〈bind targetElement="area_code" /〉
〈/DTMF〉
〈DTMF id:"extension" numDigits="7"〉
        〈bind targetElement="phone_number" /〉
〈/DTMF〉
```

この例は、いかにしてユーザが複数フィールドに入力するのを可能にするかをを示している。

【０２１７】例３：音声入力およびＤＴＭＦ入力をともに許可し、ユーザがＤＴＭＦを開始した際に音声を使用

不可にするには

```
<input type="text" name="credit_card_number" />
<prompt onBookmark="dtmf.Start(); speech.Start()"
    bargein="0">
  <bookmark name="starting" />と言うか、またはあなたのクレジット
カード番号を入力してください
</prompt>
<DTMF id="dtmf" escape="#" length="16"
interdigitTimeout="2000"
  onkeypress="speech.Stop()">
  <bind targetElement="credit_card_number" />
</DTMF>
<reco id="speech" >
  <grammar src="/grm/1033/digits.xml" />
  <bind targetElement="credit_card_number" />
</reco>
```

【０２１８】４．２　属性およびプロパティ

４．２．１　属性

・ｄｔｍｆｇｒａｍｍａｒ：必須。ＤＴＭＦ文法のＵＲＩ。

【０２１９】４．２．２　プロパティ

・ＤＴＭＦｇｒａｍｍａｒ　読み取りおよび書き込み。ストリング変換行列に対するＤＴＭＦを表すＸＭＬ　ＤＯＭ　ノードオブジェクト（ＤＴＭＦ文法とも呼ぶ）。デフォルト文法は、

```
<dtmfgrammar>
  <key value="0">0</key>
  <key value="1">1</key>
  ...
  <key value="9">9</key>
  <key value="*">*</key>
  <key value="#">#</key>
</dtmfgrammar >
```

【０２２０】・ｆｌｕｓｈ

読み取り／書き込み。起動の前に、基礎となる電話インタフェースカードのＤＴＭＦバッファを自動的にフラッシュするかどうかを示すブールフラグ。デフォルトは偽になり、タイプアヘッドを使用可能にする。

【０２２１】・ｅｓｃａｐｅ

読み取り／書き込み。ＤＴＭＦ読み取りセッションを終了するエスケープキー。エスケープキーはワンキーである。

【０２２２】・ｎｕｍＤｉｇｉｔｓ

読み取り／書き込み。ＤＴＭＦ読み取りセッションを終了させるキーストローク数。エスケープおよび長さの両方を指定した場合は、どちらかの条件を満たすとＤＴＭＦセッションが終了される。

【０２２３】・ｄｔｍｆＲｅｓｕｌｔ

読み取り専用ストリング。ユーザが入力したＤＴＭＦキーを記憶する。タイプした場合はｅｓｃａｐｅが結果に含まれる。

【０２２４】・ｔｅｘｔ

読み取り専用ストリング。空白で分離されたトークンストリングを記憶し、各トークンはＤＴＭＦ文法に従って変換する。

【０２２５】・ｉｎｉｔｉａｌＴｉｍｅｏｕｔ

読み取り／書き込み。最初のＤＴＭＦキーストロークを受け取るまでのミリ秒単位のタイムアウト期間。指定しない場合は、電話プラットフォームの内部設定になる。

【０２２６】・ｉｎｔｅｒｄｉｇｉｔＴｉｍｅｏｕｔ

読み取り／書き込み。次の（ａｄｊａｃｅｎｔ）ＤＴＭＦキーストロークまでのミリ秒単位のタイムアウト期間。指定しない場合は、電話プラットフォームの内部設定になる。

【０２２７】４．３　オブジェクトメソッド：

４．３．１　Ｓｔａｒｔ

ＤＴＭＦの割り込みを可能にし、ＤＴＭＦ読み取りセッションを開始する。

構文：Ｏｂｊｅｃｔ．Ｓｔａｒｔ（　）；

戻り値：なし

例外：なし

【０２２８】４．３．２　Ｓｔｏｐ

ＤＴＭＦを使用不可にする。ただし、ユーザが入力したキーストロークはバッファに残る。

構文：Ｏｂｊｅｃｔ．Ｓｔｏｐ（　）；

戻り値：なし

例外：なし

【０２２９】４．３．３　Ｆｌｕｓｈ

ＤＴＭＦバッファをフラッシュする。Ｆｌｕｓｈは、ＤＴＭＦセッション中には呼び出すことができない。

構文：Ｏｂｊｅｃｔ．Ｆｌｕｓｈ（　）；

戻り値：なし

例外：なし

【０２３０】４．４　イベント

４．４．１　ｏｎｋｅｙｐｒｅｓｓ

ＤＴＭＦキーを押すと発生する。これは、ＨＴＭＬコントロールから継承したデフォルトイベントを上書きする。ユーザがエスケープキーを押すと、ｏｎＫｅｙｐｒｅｓｓではなくｏｎＲｅｃイベントが発生する。

【０２３１】構文：

【０２３２】

【表１５】

インラインHTML	<DTMF onkeypress ="handler" ...>
イベントプロパティ	Object.onkeypress = handler Object.onkeypress = GetRef("handler");

【０２３３】イベントオブジェクト情報：

【０２３４】

【表１６】

バブル	なし
起動するには	タッチトーン電話のキーパッドを押す
デフォルトアクション	押されているキーを戻す

【０２３５】イベントプロパティ：イベントハンドラはプロパティを直接受け取ることはないが、ハンドラはデータについてこのイベントオブジェクトに照会を行うことができる。

【０２３６】４．４．２　ｏｎＲｅｃｏ

ＤＴＭＦセッションを終了すると発生する。このイベントは、現在のＤＴＭＦオブジェクトを自動的に使用不可にする。

【０２３７】構文：

【０２３８】

【表１７】

インラインHTML	<DTMF onReco ="handler" ...>
イベントプロパティ	Object.onReco = handler Object.onReco = GetRef("handler");

【０２３９】イベントオブジェクト情報：

【０２４０】

【表１８】

バブル	なし
起動するには	ユーザがエスケープキーを押す、またはキーストロークの回数が指定の値を満たす
デフォルトアクション	押されているキーを戻す

【０２４１】イベントプロパティ：イベントハンドラはプロパティを直接受け取ることはないが、ハンドラはデータについてこのイベントオブジェクトに照会を行うことができる。

【０２４２】４．４．３　ｏｎＴｉｍｅｏｕｔ

タイムアウトまでに、句の終了イベントを受け取らないと発生する。このイベントは、認識プロセスを自動的に停止する。

【０２４３】構文：

【０２４４】

【表１９】

インラインHTML	<DTMF onTimeout ="handler" ...>
イベントプロパティ (ECMAScript)	Object.onTimeout = handler Object.onTimeout = GetRef("handler");

【０２４５】イベントオブジェクト情報：

【０２４６】

【表２０】

バブル	なし
起動するには	指定のタイムアウト中にDTMFキーストロークが検出されない
デフォルトアクション	なし

【０２４７】イベントプロパティ：イベントハンドラはプロパティを直接受け取ることはないが、ハンドラはデータについてこのイベントオブジェクトに照会を行うことができる。

【０２４８】５　ＣａｌｌＣｏｎｔｒｏｌオブジェクト
電話音声ブラウザの電話インタフェース（呼、端末、および接続）を表す。このオブジェクトは、ＧＵＩブラウザ中のｗｉｎｄｏｗオブジェクトと同様にネイティブである。したがって、電話オブジェクトの寿命はブラウザインスタンス自体と同じである。電話用の音声ブラウザは、呼ごとに１つの電話オブジェクトをインスタンス化する。ユーザは、このオブジェクトをインスタンス化または配置しない。

【０２４９】この点で、このオブジェクトを通じて、ファーストパーティの呼の制御に関連する機能のみを示す。

【０２５０】５．１　プロパティ
・ａｄｄｒｅｓｓ
読み取り専用。ＸＭＬ　ＤＯＭ　ノードオブジェクト。実装固有。これは発呼者のアドレスである。ＰＳＴＮの場合は、ＡＮＩとＡＬＩの組合せにすることができる。ＶｏＩＰの場合、これは発呼者のＩＰアドレスになる。

【０２５１】・ｒｉｎｇＢｅｆｏｒｅＡｎｓｗｅｒ
着信呼に応答するまでの着信音の回数。デフォルトは無限。すなわち、開発者は下記のＡｎｓｗｅｒ（　）メソッドを明確に使用して、電話呼に応答しなければならない。コールセンタでＡＣＤを使用して着信電話呼をキューに入れる場合、この回数は０にセットしてよい。

【０２５２】５．２　メソッド
注：ここに示すメソッドはすべて非同期である。

【０２５３】５．２．１　Ｔｒａｎｓｆｅｒ
呼を転送する。ブラインド転送の場合、システムは転送が完了すると元の呼を終了し、システムリソースを解放する。
構文：ｔｅｌｅｐｈｏｎｅ．Ｔｒａｎｓｆｅｒ（ｓｔｒＴｅｘｔ）；
パラメータ：
・ｓｔｒＴｅｘｔ：必須。意図する受信者のアドレス。
戻り値：なし
例外：例えばエンドパーティが話中である、番号が存在しない、ファックスまたは留守番電話が応答するなど、呼の転送が失敗すると例外を投入する。

【０２５４】５．２．２　Ｂｒｉｄｇｅ
サードパーティへの転送。呼を転送すると、ブラウザはその呼に割り当てられていたリソースを解放することができる。転送した呼がｓｔｒＵＩＤを使用して戻ってきた際にセッション状態を回復するかはアプリケーション次第である。基礎となる電話プラットフォームは、戻ってきた呼を異なるブラウザに経路指定することができる。呼は、受信者がその呼を終了した際のみ戻ることができる。

【０２５５】構文：ｔｅｌｅｐｈｏｎｅ．Ｂｒｉｄｇｅ（ｓｔｒＴｅｘｔ，ｓｔｒＵＩＤ，［ｉｍａｘＴｉｍｅ］）；
パラメータ：
・ｓｔｒＴｅｘｔ：必須。意図する受信者のアドレス。
・ｓｔｒＵＩＤ：必須。現在の呼を一意に識別するセッションＩＤ。転送した呼が戻される場合、ｓｔｒＵＩＤがアドレス属性に示される。
・ｉｍａｘＴｉｍｅ：任意選択。秒単位の転送呼の最大持続時間。指定しない場合は、プラットフォームの内部値になる。
戻り値：なし
例外：なし

【０２５６】５．２．３　Ａｎｓｗｅｒ
電話呼に応答する。
構文：ｔｅｌｅｐｈｏｎｅ．Ａｎｓｗｅｒ（　）；
戻り値：なし
例外：接続がない際に例外を投入する。この場合ｏｎＡｎｓｗｅｒイベントは発生しない。

【０２５７】５．２．４　Ｈａｎｇｕｐ
電話呼を終了する。その時進行中の呼がない場合は効果を持たない。
構文：ｔｅｌｅｐｈｏｎｅ．Ｈａｎｇｕｐ（　）；
戻り値：なし
例外：なし

【０２５８】５．２．５　Ｃｏｎｎｅｃｔ
ファーストパーティへのアウトバウンドの電話呼を開始する。
構文：ｔｅｌｅｐｈｏｎｅ．Ｃｏｎｎｅｃｔ（ｓｔｒＴｅｘｔ［ｉＴｉｍｅｏｕｔ］）；
パラメータ：
・ｓｔｒＴｅｘｔ：必須。意図する受信者のアドレス。
・ｉＴｉｍｅｏｕｔ：任意選択。接続の試みを断念するまでのミリ秒単位の時間。指定しない場合は、プラットフォームの内部値になる。
戻り値：なし
例外：話中音の遭遇、あるいはファックスや留守番電話への到達を含め、呼を完了することができないと例外を投入する（注：ハードウェアがこの機能をサポートしな

い場合もある）。
【０２５９】５．２．６　Ｒｅｃｏｒｄ
ユーザオーディオをファイルに記録する。
【０２６０】構文：ｔｅｌｅｐｈｏｎｅ．Ｒｅｃｏｒｄ（ｕｒｌ，ｅｎｄＳｉｌｅｎｃｅ，［ｍａｘＴｉｍｅｏｕｔ］，［ｉｎｉｔｉａｌＴｉｍｅｏｕｔ］）；
パラメータ：
・ｕｒｌ：必須。記録された結果のＵＲＬ。
・ｅｎｄＳｉｌｅｎｃｅ：必須。無音の検出後に記録を中止するミリ秒単位の時間。
・ｍａｘＴｉｍｅｏｕｔ：任意選択。記録を行う秒単位の最大時間。デフォルトはプラットフォーム固有になる。
・ｉｎｉｔｉａｌＴｉｍｅｏｕｔ：任意選択。記録の開始時に許される無音の最大時間（ミリ秒）。
戻り値：なし
例外：記録をＵＲＬに書き込めない際に例外を投入する。

【０２６１】５．３　イベントハンドラ
電話音声ブラウザを使用するアプリケーション開発者は、以下のイベントハンドラを実装することができる。
【０２６２】５．３．１　ｏｎＩｎｃｏｍｉｎｇ（　）
音声ブラウザが着信電話呼を受信すると呼び出される。すべての開発者は、電話呼に応答する前にこのハンドラを使用して発呼者のアドレスを読み取り、カスタマイズした機能を起動することができる。
【０２６３】５．３．２　ｏｎＡｎｓｗｅｒ（　）
音声ブラウザが着信呼に応答すると呼び出される。
【０２６４】５．３．３　ｏｎＨａｎｇｕｐ（　）
ユーザが電話を切ると呼び出される。このイベントは、プログラムがＨａｎｇｕｐメソッドまたはＴｒａｎｓｆｅｒメソッドを呼び出しても自動的には発生しない。
【０２６５】５．４　例
この例は、電話セッションを操作するために呼制御イベントに結合（ｗｉｒｅ）したスクリプティングを示す。

```
<HTML>
<HEAD>
  <TITLE>ログオンページ</TITLE>
 </HEAD>
   <SCRIPT>
    var focus;
    function RunSpeech() {
        if (logon.user.value == "") {
              focus="user";
              p_uid.Start(); g_login.Start();
    dtmf.Start(); return;
         }
         if (logon.pass.value == "") {
              focus="pin";
              p_pin.Start(); g_login.Start();
    dtmf.Start(); return;
         }
         p_thank.Start(); logon.submit();
    }
    function login_reco() {
         res = event.srcElement.recoResult;
         pNode = res.selectSingleNode("//uid");
         if (pNode != null)
              logon.user.value = pNode.xml;
         pNode = res.selectSingleNode("//password");
         if (pNode != null)
              logon.pass.value = pNode.xml;
    }
    function dtmf_reco() {
         res = event.srcElement.dtmfResult;
         if (focus == "user")
              logon.user.value = res;
```

```
        else
            logon.pin.value = res;
    }
  </SCRIPT>
 <SCRIPT for="callControl" event="onIncoming">
    <!--
        // があればアドレスを読み出して、カスタマイズしたものを準備する
        callControl.Anser();
    //
  </SCRIPT>
 <SCRIPT for="callControl" event="onOffhook">
    <!--
        p_main.Start(); g_login.Start(); dtmf.Start();
    focus="user";
    //
  </SCRIPT>
 <SCRIPT for="window" event="onload">
    <!--
        if (logon.user.value != "") {
            p_retry.Start();
            logon.user.value = "";
            logon.pass.value = "";
            checkFields();
        }
    //
  </SCRIPT>
 <BODY>
   <reco id="g_login"
        onReco="login_reco(); runSpeech()"
        timeout="5000"
        onTimeout="p_miss.Start(); RunSpeech()" >
        <grammar
    src=http://kokaneel/etradedemo/speechonly/login.xml/>
        </ reco>
   <dtmf id="dtmf"
        escape="#"
        onkeypress="g_login.Stop();"
        onReco="dtmf_reco();RunSpeech()"
        interdigitTimeout="5000"
        onTimeout="dtmf.Flush();
        p_miss.Start();RunSpeech()" />
   <prompt id="p_main">あなたのユーザＩＤと個人識別番号を言ってください<
/prompt>
   <prompt id="p_uid">あなたのユーザＩＤだけを言ってください</prompt>
   <prompt id="p_pin">あなたの個人識別番号だけを言ってください</prompt>
   <prompt id="p_miss">申し訳ありませんが、聞き取れませんでした。</promp
t>
   <prompt id="p_thank">ありがとうございます。あなたの識別を確認する間お
待ちください</prompt>
   <prompt id="p_retry">申し訳ありませんが、あなたのユーザＩＤと個人識別
```

```
番号が一致しません</prompt>
  <H2>Login</H2>
<form id="logon">
        UID:   <input name="user" type="text"
        onChange="runSpeech()" />
        PIN:   <input name="pass" type="password"
        onChange="RunSpeech()" />
 </form>
 </BODY>
 </HTML>
```

【０２６６】６　ダイアログフローの制御

６．１　ＨＴＭＬおよびスクリプトを使用してダイアログフローを実装する

次の例は、入力ボックスの値を探して、入力に対して状況依存型のヘルプを提供する単純なダイアログフローの実装方法を示している。これは、ＨＴＭＬ入力機構のタイトル属性（視覚ブラウザで「ツールチップ」機構として使用される）を使用して、ヘルププロンプトの内容を形成するのを補助する。

```
<html>
   <title>状況感知型ヘルプ</title>
 <head>
   <script>    var focus;
         function RunSpeech() {
               if (trade.stock.value == "") {
                     focus="trade.stock";
                     p_stock.Start();
                     return;
               }
               if (trade.op.value == "") {
                     focus="trade.op";
                     p_op.Start();
                     return;
               }
               //.. repeat above for all fields
               trade.submit();
         }
         function handle() {
               res = event.srcElement.recoResult;
               if (res.text == "help") {
                     text = "～だけを言ってください";
                     text += document.all[focus].title;
                     p_help.Start(text);
               } else {
                     // proceed with value assignments
                     }
               }
   </script>
   </head>
 <body>
   <prompt id="p_help" onComplete="checkFileds()" />
   <prompt id="p_stock"
   onComplete="g_stock.Start()">株式銘柄を言ってください</prompt>
   <prompt id="p_op" onComplete="g_op.Start()" >売りまたは買いのどちら
```

```
をご希望ですか</prompt>
    <prompt id="p_quantity"
    onComplete="g_quantity.Start()">株式数はいくつですか</prompt>
    <prompt id="p_price"
    onComplete="g_price.Start()">価格はいくらですか</prompt>
    <reco id="g_stock" onReco="handle(); checkFields()" >
         <grammar src="./g_stock.xml" />
    </ reco >
    <reco id="g_op" onReco="handle(); checkFields()" />
         <grammar src="./g_op.xml" />
    </ reco >
    <reco id="g_quantity" onReco="handle(); checkFields()"
    />
         <grammar src="./g_quant.xml" />
    </ reco >
    <reco id="g_price" onReco="handle(); checkFields()" />
         <grammar src="./g_quant.xml" />
    </ reco >
    <form id="trade">
         <input name="stock" title="stock name" />
         <select name="op" title="buy or sell">
              <option value="buy" />
              <option value="sell" />
              </select>
         <input name="quantity" title="number of shares"
    />
         <input name="price" title="price" />
    </form>
    </body>
    </html>
```

【０２６７】６．２　ＳＭＩＬを使用するｅｃｏ要素の起動を示す。

次の例は、ＳＭＩＬ機構を使用したプロンプトおよびｒ

```
<html xmlns:t="urn:schemas-microsoft-com:time"
      xmlns:sp="urn:schemas- microsoft-
com:speech">
<head>
<style>
  .time { behavior: url(#default#time2); }
</style>
</head>
<body>
<input name="txtBoxOrigin" type="text"/>
<input name="txtBoxDest" type="text" />
<sp:prompt class="time" t:begin="0">
    出発地と行先地を言ってください
</sp:prompt>
<t:par t:begin="time.end"
t:repeatCount="indefinitely"
  <sp:reco class="time" >
     <grammar src="./city.xml" />
```

```
        <bind targetElement="txtBoxOrigin"
         value="//origin_city" />
        <bind targetElement="txtBoxDest"
         test="/sml/dest_city[@confidence $gt$ 40]"
         value="//dest_city" />
      </sp:reco>
    </t:par>
    </body>
    </html>
```

【０２６８】７．ＳＭＥＸ（メッセージ）要素／オブジェクト

ＳＭＥＸは、Ｓｉｍｐｌｅ　Ｍｅｓｓａｇｉｎｇ　ＥＸｃｈａｎｇｅ／ＥＸｔｅｎｓｉｏｎの略語であるが、これは、クライアントデバイスのプラットフォーム上の外部コンポーネントまたはアプリケーションと通信するオブジェクトである。これは、タグ名＜ｓｍｅｘ＞を有する要素として、ＸＭＬまたはそれに類似のマークアップベースのドキュメント中に埋め込むことができる。このメッセージングオブジェクトの使用例には、ロギングおよび電話制御を含むことができる。このオブジェクトは、メッセージングを通じて新しい機能を追加することを可能にすることから、マークアップベースの認識およびプロンプティング（ｐｒｏｍｐｔｉｎｇ）の拡張性を表す。

【０２６９】インスタンスを生成すると、このオブジェクトは、その構成パラメータまたは属性指定を通じて、プラットフォームコンポーネントまたはアプリケーションとの非同期のメッセージ交換経路を確立するように指示を受ける。このオブジェクトはストリングプロパティを有し、そのプロパティが割り当て動作（すなわちｌｖａｌｕｅ）を受ける対象である場合には、必ずその内容がプラットフォームコンポーネントまたはアプリケーションに送られる。同様に、このオブジェクトは、プラットフォームコンポーネントまたはアプリケーションから受け取ったメッセージを保持する、ＸＭＬ　ＤＯＭノードタイプのプロパティも有する。このメッセージオブジェクトは、プラットフォームメッセージを受け取ると必ずイベントを送る。このオブジェクトは、その基本動作が非同期なので、アプリケーション開発者がタイムアウト設定を操作するための内蔵クロックも有する。

【０２７０】メッセージまたはｓｍｅｘオブジェクトは、通信手段にとってアグノスティック（ａｇｎｏｓｔｉｃ）である。しかし、一実施形態では、ｓｍｅｘオブジェクトは、通常のＸＭＬやマークアップ要素と同じ寿命を有する。すなわち、ｓｍｅｘオブジェクトは、それをホストするドキュメントをアンロードすると消滅する。多くのケースでは、ｓｍｅｘオブジェクトはアンロードされると自動クリーンアップを実行し、通信リソースを解放することができるが、マークアップページ間で永続的な通信リンクが望ましい使用事例（例えば呼の制御など）もありうる。そのような事例のために、このアーキテクチャでは、割り振られたリソースを解放する（例えばソケットを閉じるなど）責任をアプリケーション開発者に課す。

【０２７１】ｓｍｅｘオブジェクトは、メッセージのフォーマット（スキーマ）についてはニュートラルである。実施形態によっては、既存の標準的なメッセージフォーマット（例えばＳＴＰまたはＣＣＸＭＬで使用するものなど）をまず優先して、実装者がいくつかの基本的スキーマをサポートすることを必要とするのが望ましい場合もある。基本的に、このアーキテクチャは、プラットフォーム開発者およびアプリケーション開発者の両者が、ＸＭＬあるいはそれに類似のマークアップの規格化された拡張性を最大限に活用して、一方では相互操作性を失うことなく他の機能を導入することを可能にする。

【０２７２】

例１：ロギングオブジェクトとしてのｓｍｅｘの使用

```
<smex_id="logServer">
     <param name="d:server"
xmlns:d="urn:Microsoft.com/COM">
          <d:protocol>DCOM</d:protocol>
     <d:clsid>20930930293020293209420984320 98</d:clsid>
     <d:iid>0903859304903498530985309094803</d=iid>
     </param>
</smex>
<listen...>
     ...//ｒｅｃｏ結果を入力フィールドにバインドする他のディレクティブ
.......  <bind targetElement="logServer"
```

```
        targetAttribute="sent"
                value="*[@log SgeS 3]/>
        </listen>
```

【０２７３】この例は、ＣＯＭオブジェクトをそのクラスＩＤおよびインタフェースＩＤとともに使用して、ロギング機構を実現する仕組みを示している。音声開発者は、関連するＳＭＬノードにロギングするための当該レベルを示す属性「ｌｏｇ」を付加する。上の例では、アプリケーション開発者が、単一のバインドディレクティブを使用することにより、３を超えるか、または３に等しいログ値を有するノードすべてにロギングすることを選択している。この例は、ダウンレベルブラウザでもアップレベルブラウザでも機能する。

【０２７４】この例はまた、ｓｍｅｘオブジェクトがプラットフォームメッセージを認識ドキュメントに伝達する役割を負うような混乱状態がない限り、あるページが、同じプラットフォームコンポーネントと通信する複数のｓｍｅｘオブジェクトを含むことが可能であることも示すものである。上の例は、あるコンポーネントが複数のインタフェースを実装することができ、それぞれのインタフェースがそれ自体のｓｍｅｘまたはメッセージ経路を有することを示唆している。これと同じ論議は、複数のポートをリッスンするＴＣＰサーバにも当てはまる。

【０２７５】

例２：着信呼のアドレスの読み取り

```
<input type="text" id="remote"/>
<input type= "text" id="transfer"/>
<input type= "text" id= "local"/>
<input type= "hidden" id = "session_id "/>
.....
<smex id="telephone"  sent= "start_listening">
      <param name="server">http://tel-svr/whatever</param>
      <bind targetElement = "session_id" value="//sid"/>
      <bind targetElement= "remote"
value="//remote_addr"/>
      <bind targetElement="transfer"
value="//transfer_addr/>
      <bind targetElement = "local"
value="//local_addr/>
    ....
</smex>
```

【０２７６】この例は、どのようにバインドディレクティブを使用して、受信メッセージを処理できるかを示している。この例では、着信呼のメッセージが、下位要素のｒｅｍｏｔｅ＿ａｄｄｒ、ｔｒａｎｓｆｅｒ＿ａｄｄｒ、およびｌｏｃａｌ＿ａｄｄｒを有するものと想定しており、その内容はそれぞれ着信呼のリモートアドレス、転送アドレス、およびローカルアドレスを表す。

【０２７７】この例では、ＨＴＴＰに基づくコネクションレスプログラミングを使用して電話サーバと通信する。この場合の電話サーバは、複数のブラウザインスタンスと通信するように設計されており、したがって、各クライアントは、アプリケーションの開始時にサーバから割り当てられる一意のＩＤによって自らを識別しなければならない。この例では、これはサーバに「ｓｔａｒｔ＿ｌｉｓｔｅｎｉｎｇ」メッセージを送信することによって実現する。この例では、セッションＩＤを隠しフィールドに記憶し、それをウェブサーバに送信して、アプリケーションの次のページに渡すことができるが、セッション状態の管理には他の技術（例えばクライアントサイドのクッキー）も使用することができる。ｒｅｃｏの場合と同様に、あらゆるプラットフォームメッセージについてすべてのバインドディレクティブが実行されるとは限らない。上の例は、着信電話呼がある際に一意のＩＤのみを受信することは示唆していない。

【０２７８】７．１　プロパティ

ｓｍｅｘオブジェクトは以下のプロパティを有することができるが、初期値指定のための属性としても機能することができるのは、読み取り／書き込みのプロパティだけである。

【０２７９】・ｓｅｎｔ：読み取り／書き込み。プラットフォームコンポーネントに送信するメッセージに対応するストリング。ｌｖａｌｕｅとしてｓｅｎｔを使用する場合は、必ずその内容をディスパッチする。このプロパティをｒｖａｌｕｅとして使用する場合、あるいはこのプロパティにヌルオブジェクトを割り当てる場合には効果がない。

【０２８０】・ｒｅｃｅｉｖｅｄ：読み取り専用。受信メッセージを表すＸＭＬ　ＤＯＭノードデータ。このメ

ッセージは、次のｏｎＲｅｃｅｉｖｅイベントが送ることのできる状態になるまで、ｒｖａｌｕｅとして使用することができる。

【０２８１】・ｔｉｍｅｒ：読み取り／書き込み。タイムアウトイベントをトリガするまでの時間を表すミリ秒単位の数。クロックは、このプロパティに正の値が割り当てられると刻時を開始する。この値は、カウントダウンの進行中に変更することができる。ゼロまたは負の値にすると、タイムアウトイベントをトリガせずにクロックを停止する。デフォルトは０、すなわちタイムアウトなしである。

【０２８２】・ｓｔａｔｕｓ：読み取り専用。オブジェクトの最近のステータスを表す整数。可能な値は、０、－１、および－２であり、それぞれ、正常、タイムアウトの終了、およびプラットフォームとの通信を確立できない、あるいは通信の中断を意味する。受信されるプロパティを通じて、プラットフォーム固有のエラーメッセージを伝達するとよい。エラーメッセージの伝達が成功した場合、ステータスコードは０になる。

【０２８３】７．２　イベント

このオブジェクトは以下のイベントを有する。

【０２８４】・ｏｎＲｅｃｅｉｖｅ：このイベントは、プラットフォームメッセージが到着すると送られる。バインド要素によって宣言されたディレクティブがある場合には、このイベントを発生させる前にそのディレクティブを先に評価する。イベントを送る前に、受け取ったプロパティを更新する。

【０２８５】・ｏｎＥｒｒｏｒ：このイベントは、タイムアウトが経過したとき、あるいは通信リンクエラーに遭遇したときに送られる。このイベントを送る際、上記のように、ステータスプロパティをそれに対応するエラーコードによって更新する。

【０２８６】７．３　子要素

ある要素の形を仮定するとき、ｓｍｅｘは以下の子要素を有することができる。

・ｂｉｎｄ：ディレクティブを受信メッセージに作用させる点を除いては、ｒｅｃｏの場合と同様。

・ｐａｒａｍ：ｒｅｃｏの場合と同様。ｓｍｅｘオブジェクトのプラットフォーム固有パラメータを提供する。各ｐａｒａｍ要素は、「ｎａｍｅ」属性を使用して名前をつけることができ、ｐａｒａｍ要素の内容がそのパラメータの値になる。一実施形態では、この要素は、ネームスペースの標準的なＸＭＬ属性とＸＭＬデータタイプ宣言を理解しているべきである。

【０２８７】７．４　その他の補足説明

ロギング機能のためにＳＭＥＸを拡張する簡潔な方法の１つが以下である。

```
〈smex id="logServer"...〉 ... 〈/smex〉
〈script〉 function logMessage(logClass, message){
    logServer.sent = logClass + "|" + message;
} 〈/script〉
```

これは、実際に、その振る舞いを個別設定することのできる（グローバル）関数でこのオブジェクトを拡張している。上の例では、ＩＤとメッセージの間にフィールド区切り文字「｜」を挿入するようにロギング関数をプログラムしている。

【０２８８】グローバル関数を好まない者は、ＥＣＭＡＳｃｒｉｐｔの「ｐｒｏｔｏｔｙｐｅ」プロパティを使用して、この関数をオブジェクトメソッドとして付加することができる。例えば、

```
<smex id = "logServer" onload="addFunction( )">
</smex>
<script>
    function my_logMessage(logClass, message) {
        logServer. sent = logClass + "¦" + message;
        }
        function addFunction( ){
            logServer. prototype. logMessage=
my_logMessage;
      }
</script>
```

よりオブジェクト指向的な方式でこの関数を参照することができる。

```
logServer. logMessage(RECO_LOG_ERROR, "My message");
```

上記の例のように拡張を機能させるために、ｓｍｅｘオブジェクトの実装者にはより多くの作業が要求されるが、すべての必要な機構はすでに確立された規格であることに留意されたい。

【０２８９】

【発明の効果】以上、説明したように、本発明によれば、インターネットなどのサーバ／クライアントアーキテクチャで音声認識を提供するのに使用されるウェブ対応音声認識用サーバは、統一したアーキテクチャを備えることが可能となる。

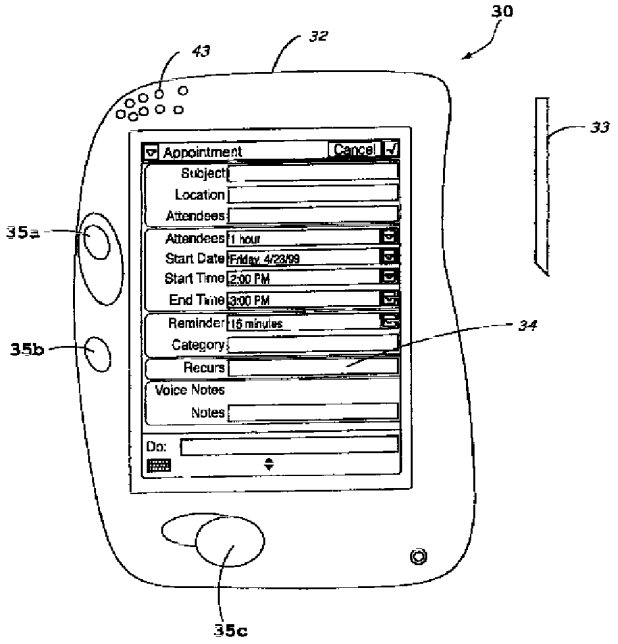
【図面の簡単な説明】

【図１】本発明の実施形態の、コンピューティングデバ

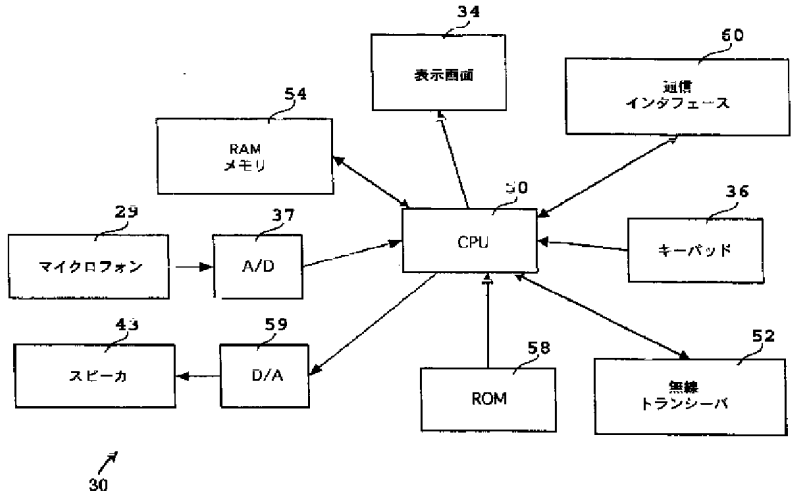
イスの動作環境の第１の実施形態の平面図である。
【図２】本発明の実施形態の、図１のコンピューティングデバイスのブロック図である。
【図３】本発明の実施形態の、電話機の平面図である。
【図４】本発明の実施形態の、汎用コンピュータのブロック図である。
【図５】本発明の実施形態の、クライアント／サーバシステムのアーキテクチャのブロック図である。
【図６】本発明の実施形態の、クレジットカード情報を得るための表示の図である。
【図７】本発明の実施形態の、クライアントで実行することのできるマークアップ言語のページの図である。
【図８】本発明の実施形態の、ディスプレイおよび音声認識機能を有するクライアントで実行することのできるマークアップ言語の例示的ページの図である。
【図９】本発明の実施形態の、音声レンダリングのみを用い、システム主導型で、クライアントで実行できるマークアップ言語の例示的ページの図である。
【図１０】本発明の実施形態の、音声レンダリングのみを用い、システム主導型で、クライアントで実行できるマークアップ言語の例示的ページの図である。
【図１１】本発明の実施形態の、音声レンダリングのみを用い、混合主導型で、クライアントで実行できるマークアップ言語の例示的ページの図である。
【図１２】本発明の実施形態の、音声レンダリングのみを用い、混合主導型で、クライアントで実行できるマークアップ言語の例示的ページの図である。
【図１３】本発明の実施形態の、サーバサイドのプラグインモジュールによって実行することのできる例示的スクリプトの図である。
【図１４】本発明の実施形態の、認識サーバの第１の動作モードを図式的に示す図である。
【図１５】本発明の実施形態の、認識サーバの第２の動作モードを図式的に示す図である。
【図１６】本発明の実施形態の、認識サーバの第３の動作モードを図式的に示す図である。
【図１７】本発明の実施形態の、スクリプティングを用いないクライアントで実行することのできる宣言的マークアップ言語の例示的ページの図である。
【図１８】本発明の実施形態の、スクリプティングを用いないクライアントで実行することのできる宣言的マークアップ言語の例示的ページの図である。
【符号の説明】
２９、１８３　マイクロフォン
３０　データ管理デバイス（モバイルデバイス、クライアント）
３２　筐体
３３　スタイラス
３４　ディスプレイ
３５ａ、３５ｂ、３５ｃ　ボタン
３６　キーパッド
３７、５９　Ａ／Ｄ変換器
４３、１８７　スピーカ
５０　ＣＰＵ
５２　無線トランシーバ
５４、１５２　ＲＡＭ
５８、１５１　ＲＯＭ
６０　通信インタフェース
８０　電話機
８２　ディスプレイ
８４　キーパッド
１２０　汎用コンピュータ
１４０　プロセッサ
１４１　システムバス
１５０　システムメモリ
１５３　ＢＩＯＳ
１５４、１６４　オペレーティングシステム
１５５、１６５　アプリケーションプログラム
１５６、１６６　プログラムモジュール
１５７、１６７　プログラムデータ
１６１　ハードディスクドライブ
１６０、１７０　インタフェース
１６０　取外し不能不揮発性メモリインタフェース
１７０　リムーバル不揮発性メモリインタフェース
１７１　磁気ディスクドライブ
１７２　磁気ディスク
１７５　光ディスクドライブ
１７６　光ディスク
１８０　ユーザ入力インタフェース
１８１　ポインティングデバイス
１８２　キーボード
１８４　モニタ
１８５　ビデオインタフェース
１８６　プリンタ
１８８　出力周辺インタフェース
１９０　ネットワークインタフェース
１９１　ＬＡＮ
１９２　モデム
１９３　ＷＡＮ
１９４　リモートコンピュータ
１９５　リモートアプリケーションプログラム
２００　アーキテクチャ
２０２　ウェブサーバ
２０４　認識サーバ
２０５　ネットワーク
２０７　専用回線
２０８　電話網
２１０　ゲートウェイ
２１１　レコグナイザ
２１２　電話音声ブラウザ

２１４　メディアサーバ
２１６　音声ブラウザ
２５０、２５２、２５４　フィールド
２６０、２７０、３００　本体部分
２６２、２７２、３０２　スクリプト部分
２６４　提出ボタン
２８０、２８２、２８４、４０５　コード部分
２８１　スケジュール
２８３、３０５　認識の開始
２８５　音声の検出
２８７　音声の終了
２８９、２９１、２９３、２９５、２９７、２９９、３０１、３０３、３０５
属性（期間、イベント）
２９０、３０３　コントロール
３０７　音声変換システム
３０９　パーサ
３２０、３２４　モジュール

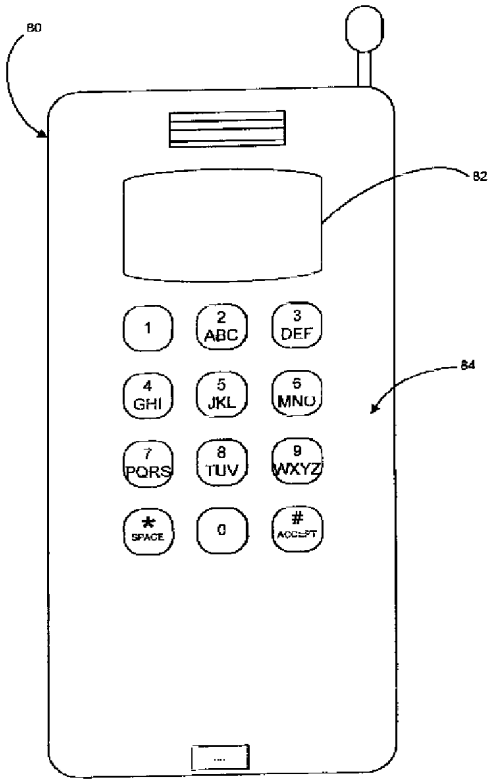
【図１】



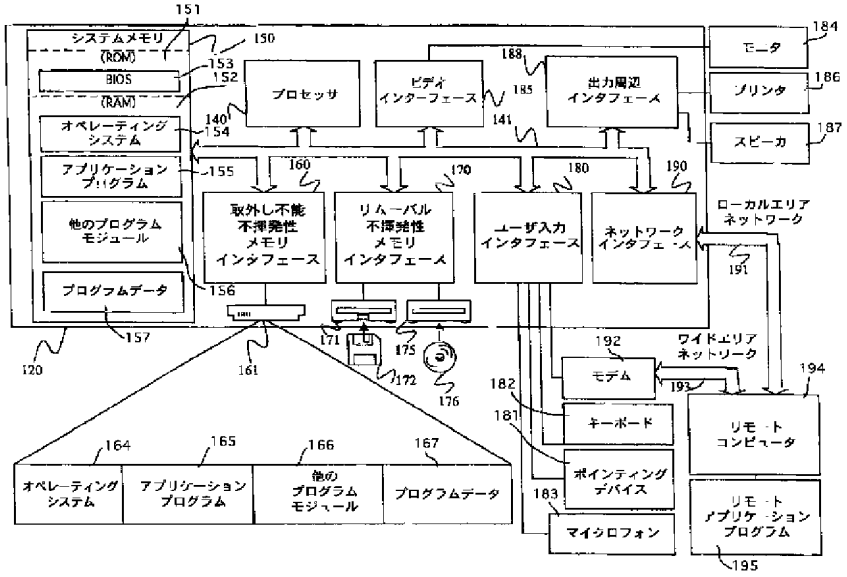
【図２】



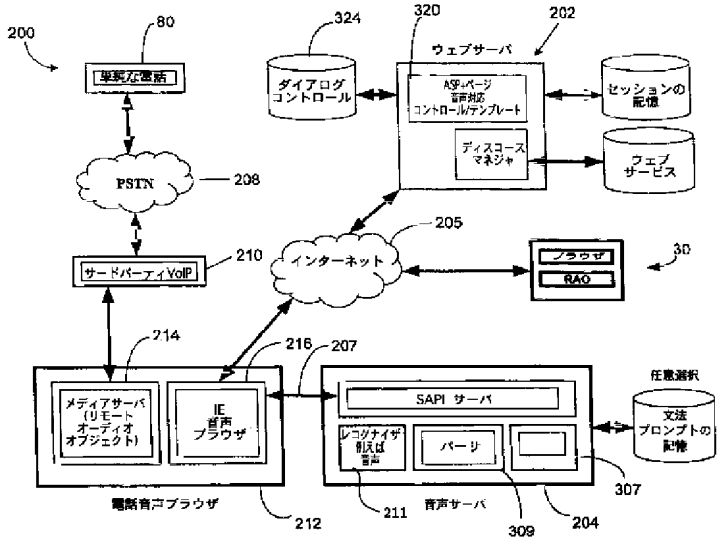
【図３】



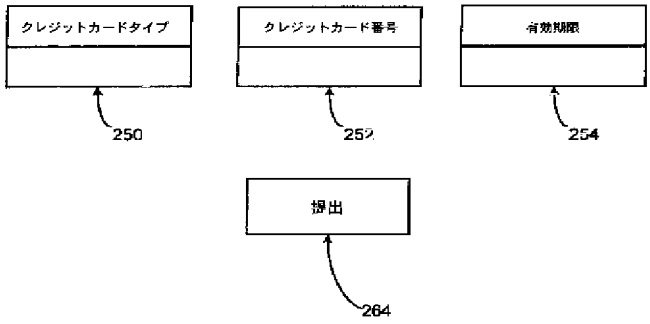
【図４】


システムメモリ
(ROM)
BIOS
(RAM)
オペレーティング システム
アプリケーション プログラム
他のプログラム モジュール
プログラムデータ
プロセッサ
ビデオ インターフェース
出力周辺 インタフェース
モニタ
プリンタ
スピーカ
取外し不能 不揮発性 メモリ インタフェース
リムーバル 不揮発性 メモリ インタフェース
ユーザ入力 インタフェース
ネットワーク インタフェース
ローカルエリア ネットワーク
ワイドエリア ネットワーク
モデム
キーボード
ポインティング デバイス
マイクロフォン
リモート コンピュータ
リモート アプリケーション プログラム
オペレーティング システム
アプリケーション プログラム
他の プログラム モジュール
プログラムデータ

【図５】


200
80
単純な電話
PSTN
208
サードパーティVoIP
210
ダイアログ コントロール
324
320
ウェブサーバ
202
ASPページ 音声対応 コントロール/テンプレート
ディスコース マネジャ
セッションの記憶
ウェブ サービス
インターネット
205
ブラウザ
RAO
30
メディアサーバ (リモート オーディオ オブジェクト)
IE 音声 ブラウザ
214
216
207
電話音声ブラウザ
212
SAPI サーバ
レコグナイザ 例えば 音声
パーサ
音声サーバ
211
309
204
307
任意選択
文法 プロンプトの記憶

【図6】



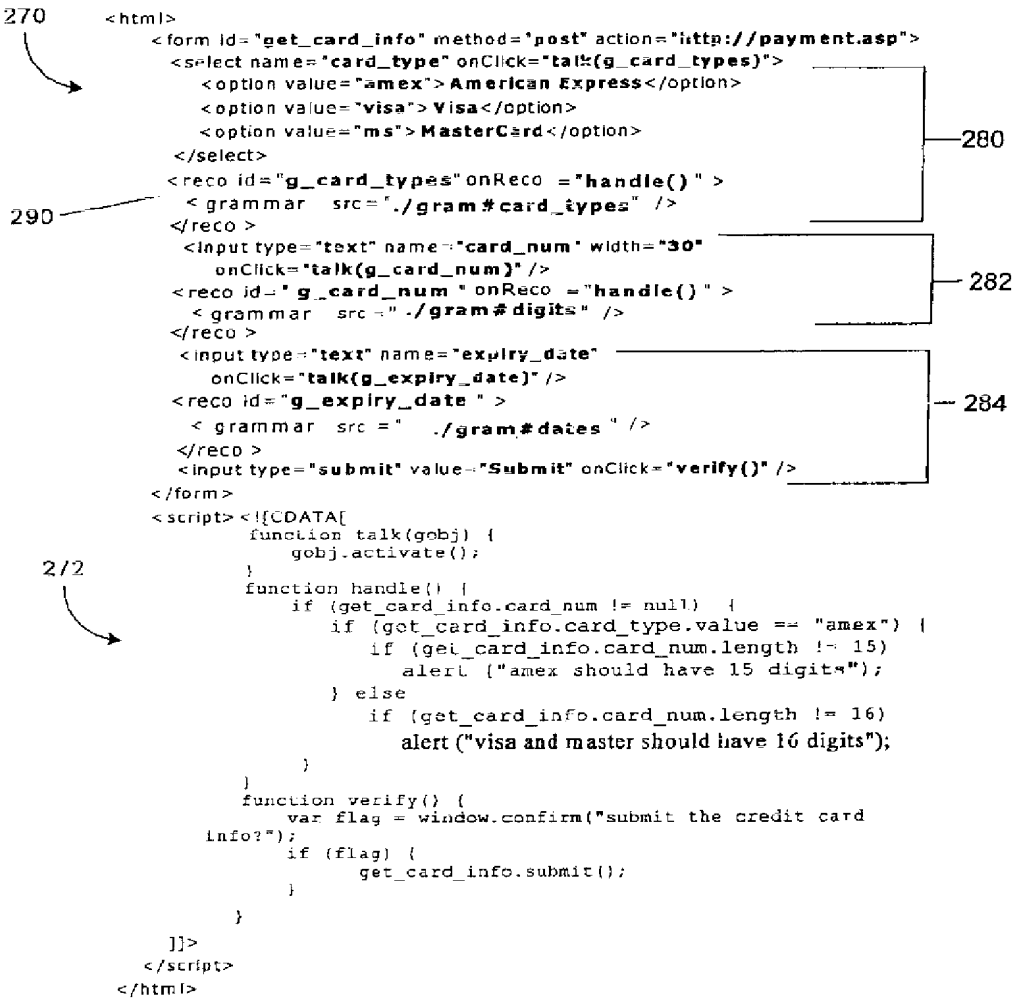
【図7】

```
<html>
  <form id="get_card_info" method="post" action="http://payment.asp">
    <select name="card_type">
       <option value="amex">American Express</option>
       <option value="visa">Visa</option>
       <option value="ms">MasterCard</option>
    </select>
    <input type="text" name="card_num" width="30"
       onChange="handle()" />
    <input type="text" name="expiry_date" />
    <input type="submit" value="Submit" onClick="verify()" />
  </form>
  <script><![CDATA[
          function handle() {
             if (get_card_info.card_type.value == "amex") {
                if (get_card_info.card_num.length != 15)
                   alert ("amex should have 15 digits");
             } else
                if (get_card_info.card_num.length != 16)
                   alert ("visa and master should have 16 digits");
          }
          function verify() {
             var flag = window.confirm("submit the credit card info?");
             if (flag)
                get_card_info.submit();
          }
    ]]>
  </script>
</html>
```

260

262

【図8】




```
<html>
  <form id="get_card_info" method="post" action="http://payment.asp">
    <select name="card_type" onClick="talk(g_card_types)">
      <option value="amex">American Express</option>
      <option value="visa">Visa</option>
      <option value="ms">MasterCard</option>
    </select>
    <reco id="g_card_types" onReco ="handle()" >
      <grammar src="./gram#card_types" />
    </reco >
    <input type="text" name="card_num" width="30"
      onClick="talk(g_card_num)" />
    <reco id=" g_card_num " onReco ="handle()" >
      <grammar src=" ./gram#digits" />
    </reco >
    <input type="text" name="expiry_date"
      onClick="talk(g_expiry_date)" />
    <reco id="g_expiry_date " >
      <grammar src=" ./gram#dates " />
    </reco >
    <input type="submit" value="Submit" onClick="verify()" />
  </form>
  <script><![CDATA[
        function talk(gobj) {
            gobj.activate();
        }
        function handle() {
            if (get_card_info.card_num != null) {
                if (get_card_info.card_type.value == "amex") {
                    if (get_card_info.card_num.length != 15)
                        alert ("amex should have 15 digits");
                } else
                    if (get_card_info.card_num.length != 16)
                        alert ("visa and master should have 16 digits");
            }
        }
        function verify() {
            var flag = window.confirm("submit the credit card
    info?");
            if (flag) {
                get_card_info.submit();
            }
        }
    ]]>
  </script>
</html>
```

【図9】

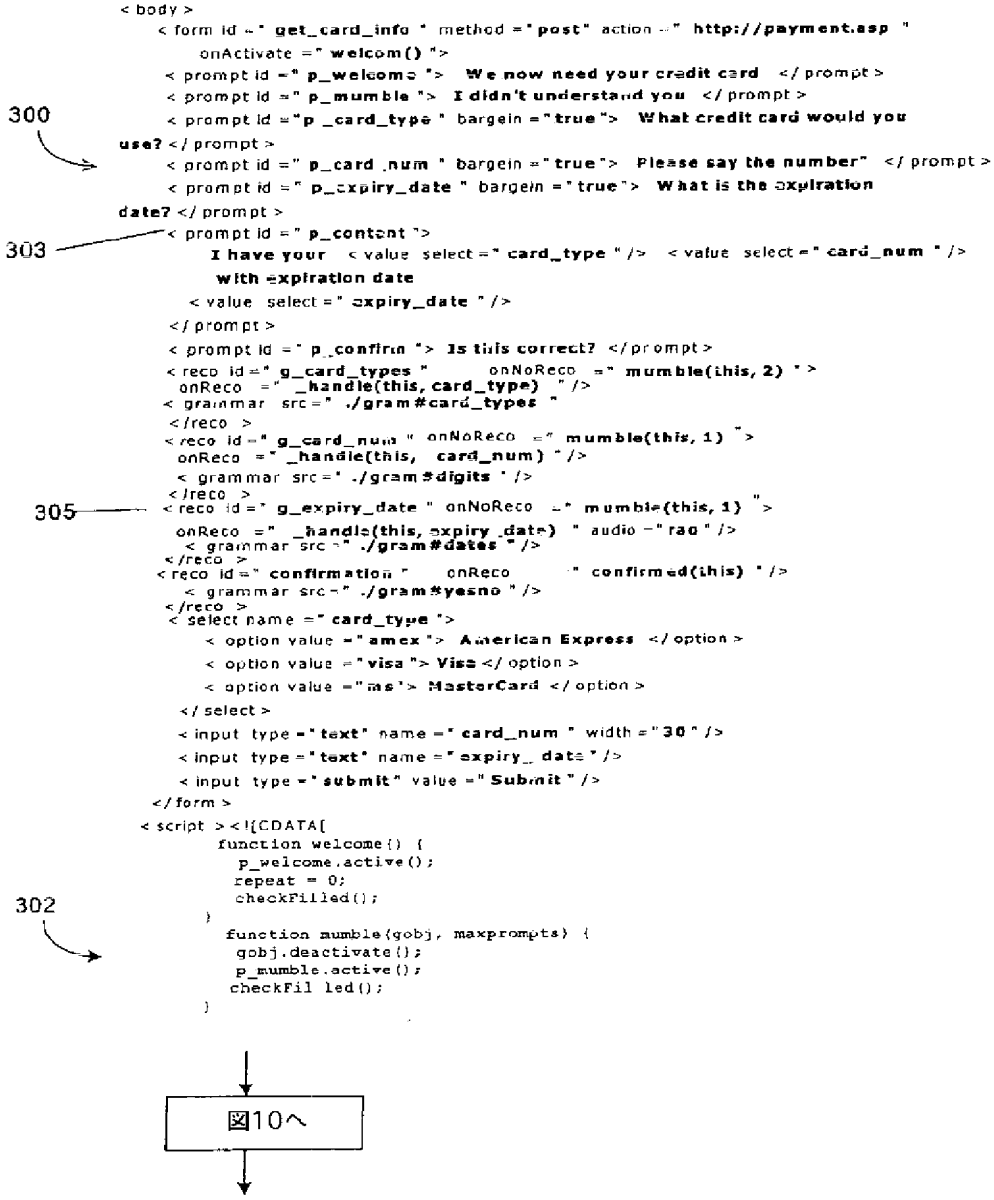

300
303
305
302
< body >
< form id =" get_card_info " method =" post " action =" http://payment.asp "
onActivate =" welcom() ">
< prompt id =" p_welcome "> We now need your credit card </ prompt >
< prompt id =" p_mumble "> I didn't understand you </ prompt >
< prompt id =" p _card_type " bargein =" true "> What credit card would you use? </ prompt >
< prompt id =" p_card_num " bargein =" true "> Please say the number" </ prompt >
< prompt id =" p_expiry_date " bargein =" true "> What is the expiration date? </ prompt >
< prompt id =" p_content ">
I have your < value select =" card_type " /> < value select =" card_num " />
with expiration date
< value select =" expiry_date " />
</ prompt >
< prompt id =" p_confirm "> Is this correct? </ prompt >
< reco id =" g_card_types " onNoReco =" mumble(this, 2) "
onReco =" _handle(this, card_type) " />
< grammar src =" ./gram#card_types "
</reco >
< reco id =" g_card_num " onNoReco =" mumble(this, 1) "
onReco =" _handle(this, card_num) " />
< grammar src =" ./gram#digits " />
</reco >
< reco id =" g_expiry_date " onNoReco =" mumble(this, 1) "
onReco =" _handle(this, expiry_date) " audio =" rao " />
< grammar src =" ./gram#dates " />
</reco >
< reco id =" confirmation " onReco =" confirmed(this) " />
< grammar src =" ./gram#yesno " />
</reco >
< select name =" card_type ">
< option value =" amex "> American Express </ option >
< option value =" visa "> Visa </ option >
< option value =" ms "> MasterCard </ option >
</ select >
< input type =" text " name =" card_num " width =" 30 " />
< input type =" text " name =" expiry_ date " />
< input type =" submit " value =" Submit " />
</ form >
< script > <![CDATA[
function welcome() {
p_welcome.active();
repeat = 0;
checkFilled();
}
function mumble(gobj, maxprompts) {
gobj.deactivate();
p_mumble.active();
checkFil led();
}

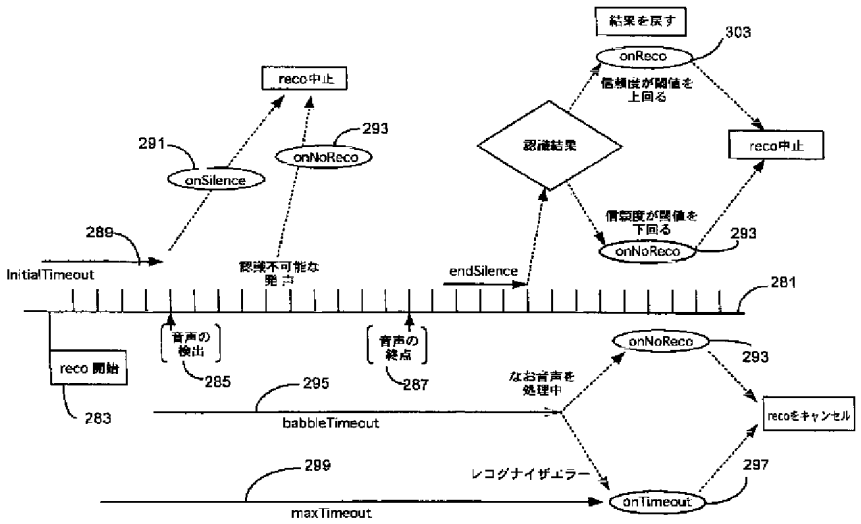
図10へ

【図10】

```
        function _handle() {
          handle();
          checkFilled();
        }
        function checkFilled() {
          if (card_type.value == "") {
                p_card_type.active();
                g_card_types.activate(); return;
          }
          if (card_num.value =="") {
                p_card_num.active(); g_card_num.acti   vate(); return;
          }
          if (expiry_date.value == "") {
                 p_expiry_data.active(); g_expiry_date.activate(); return;
          }
           p_content.activate();
          p_confirm.activate();
          confirmation.activate();
      }
        function confirmed(gobj) {
                if (gobj.recogRes.text == "yes")
                        get_card_info.submit();
        }
        //
        // user codes start here
        //
        function handle() {

          if (get_card_info.card_type == "amex") {
             if (get_card_info.card_num.length != 15) {
                prompt.speak ("amex should have 15 digits");
                get_card_info.card_num = "";
             }
          } else
             if (get_card_info.card_num.length != 16)  {
                prompt.speak ('visa and master should have 16 digits");
                get_card_info.card_num = "";
             }
        }
    ]]>
  </script>
</body>
```

【図14】



【図11】

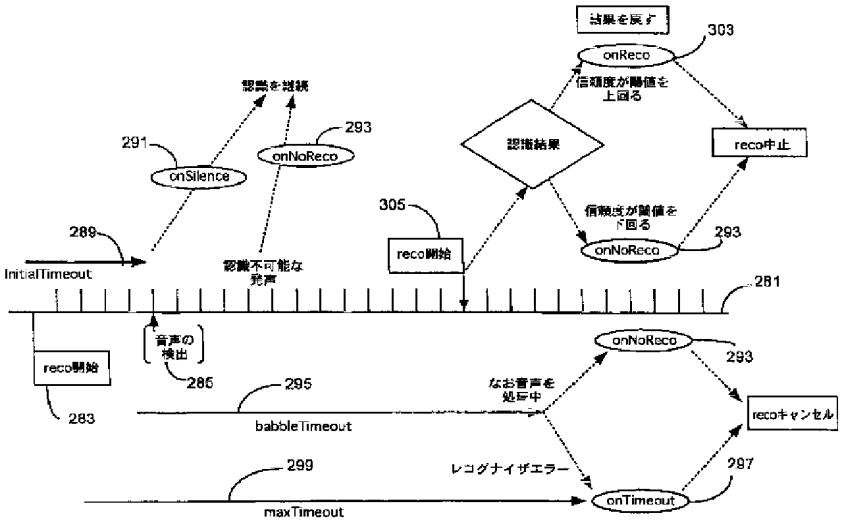
```
<body>
    <form id="get_card_info" method="post" action="http://payment.asp"
      onactivate="welcome()">
      <prompt id="p_welcome">We now need your credit card</prompt>
      <prompt id ="p_mumble">I didn't understand you</prompt>
      <prompt id="p_card_type" bargein="true">What credit card would you
use?</prompt>
      <prompt id="p_card_num" bargein="true">Please say the number</prompt>
      <prompt id="p_expiry_date" bargein="true">What is the expiration
date?</prompt>
      <prompt id="p_content">
        I have your <value select="card_type" /><value select="card_num" />
  with expiration date <value select="expiry _date" />
      </prompt>
      <prompt id="p_confirm">Is this correct?</prompt>
      <reco id="g_get_card_info" onReco="_handle()" onNoReco ="mumble(this)">
        <grammar src ="./gram/getPayment" />
        <bind target="card _type" value="/card/type" />
        <bind target="card_num" value="/card/number" />
405 ——  <bind target="expiry_date" value="/card/expir_date" />
      </reco>
      <reco id="confirmation" onReco ="confirmed(this) onNoReco="mumble(this)" />
        <grammar src ="./gram#yesno"/>
      </reco >
      <select name="card_type">
        <option value="amex">American Express</option>
        <option value="visa">Visa</option>
        <option value="ms">MasterCard</option>
      </select>
      <input type="text" name="card_num" width="30" />
      <input type="text" name="expiry_date" />
      <input type="submit" value="Submit" />
    </form>
    <script><![CDATA[
        function welcome() {
          p_welcome.active();
          repeat = 0;
          checkFilled();
        }
        function mumble(gobj) {
          gobj.deactivate();
          p_mumble.active();
          checkFilled();
        }
        function _handle() {
          handle();
          checkFilled();
        }
```

図12へ

【図12】

```
        function checkFilled()   {
          if (card_type.value == "") {
                p_card_type.active();; do_field.activate(); return;
          }
          if (card_num.value =="") {
                p_card_num.active(); do_field.activate(); return;
          }
          if (expiry_date.value == "") {
                p_expiry_data.active(); do_field.activate(); return;
          }
          p_content.activate();
          p_confirm.activate();
          confirmation.activate();
        }
        function confirmed(gobj) {
          if (gobj.recogRes.text == "yes")
                get_card_info.submit(gensml());
        }
        //
        // user codes start here
        //
          function handle() {
          if (field == get_card_info.card_num) {
             if (get_card_info.card_num.length != 15) {
                prompt.speak ("amex should have 15 digits");
                get_card_info.card_num = "";
             }
          } else
             if (get_card_info.card_num.length != 16)  {
                prompt.speak ("visa should have 15 digits");
                get_card_info.card_num = "";
             }
      }
          function gensml() {
                str = '<sml><credit_card type="';
                str += card_type.value; str += '"><number>';
                str += card_number.value; str += '</number><expire>';
                str += expiry_date.value; str +=
'</expire></credit_card></sml>';
                return str;
      }
    ]]>
  </script>
</body>
```

【図15】



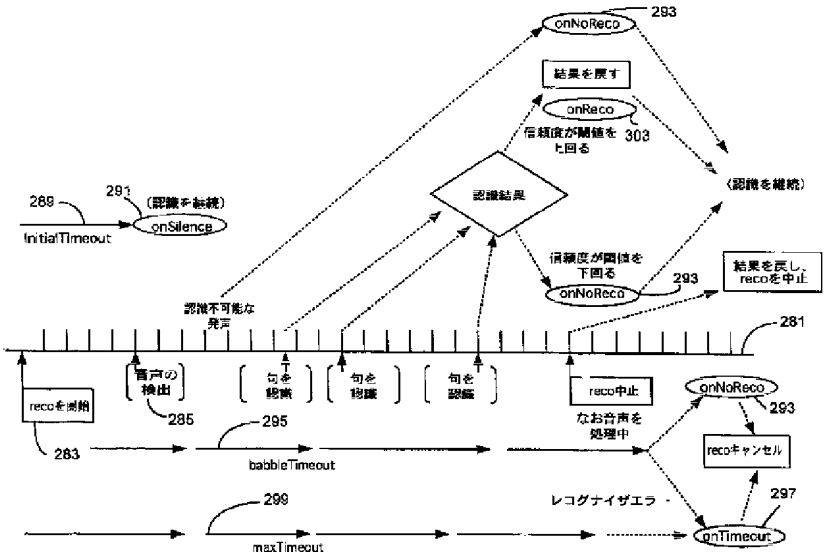
【図13】

ASPページの一例

```
<%@ Page language="Jscript" AutoEventWireup="false" Inherits="Credit.Transaction" %>
<html><head>
<!-- ASPX page for both voice only & multimodal credit card example -->
<script>
function handle() {
         if (field == get_card_info.card_num) {
            if (get_card_info.card_num.length != 15) {
              prompt.speak ("amex should have 15 digits");
              get_card_info.card_num = "";
            }
         } else
            if (get_card_info.card_num.length != 16)  {
              prompt.speak ("visa should have 15 digits");
              get_card_info.card_num = "";
            }
       }
function genSML() {
       str = '<SML><credit_card><card_type>';
       str += card_type.value; str += '</card_type><number>';
       str += card_number.value; str += '</number><expire>';
       str += expiry_date.value; str += '</expire></credit_card></SML>';
       return str;
}
</script>
<script runat="server">
function Page_Load (obj, args) {
       if (PostBack) {
          validator = new System.Speech.SMLValidator("./CreditSDL.xml");
          dsml = validator.Evaluate(args);
          Navigate(ChoosePage(dsml));
       } else {
         // initialize fields with args
       }
}
</script>
</head>
<body>
<speech:form id="get_card_info" style="system_initiative"
       prompt="./prompt/getPayment" onsubmit="genSML()">
  <speech:choice name="card_type" prompt="What credit card would you use?"
       grammar="./gram#card_types" onPhraseFinish="handle()"
       <option>American Express</option>
       <option>Visa</option>
       <option>Mastercard </option>
  </speech:choice>
  <speech:textbox name="card_number" prompt="Please say the number"
       grammar="./gram#digits" onPhraseFinish="handle()">
  <speech:textbox name="expiry_date" prompt="What is the expiration date? "
       grammar="./gram#dates" onPhraseFinish="handle()"/>

</speech:form>
</body>
</html>
```

【図16】



【図17】

```
<html>
<body>
<!-- the data section -->
350   <form id="get_drink">
        <input name="drink" />
        <input type="radio" name="cream"/>
        <input type="radio" name="sugar"/>
        <input name="uid" type="hidden"/>
      </form>
<!-- The speech section -->
352   <prompt id="welcome">Welcome, caller! </prompt>
      <prompt id="ask"> Do you want coke, coffee, or orange juice?
      </prompt>
      <prompt id="confirm"> I heard <value href="drink"/>. Is this
           correct? </prompt>
      <prompt id="thanks"> Thank you. Please wait when I get it for
           you. </prompt>
      <prompt id="retry"> Okay, let's do this again </prompt>
      <prompt id="reprompt"> Sorry, I missed that. </prompt>
      <prompt id="cream_sugar">Do you want cream or sugar with your
           coffee?</prompt>

    <reco id ="reco_drink"><grammar src="./drinktypes"/>
354 364 <bind test="/[@confidence $le$ 10]"
        366 targetElement="reprompt" targetMethod="start"
        368 targetElement="ask" targetMethod="start"
        370 targetElement="reco_drink" targetMethod="start"/>
      <bind test="/drink/coffee[@confidence $gt$ 10]"
        372 targetElement="drink" value="/drink"
        374 targetElement="cream_sugar" targetMethod="start"
        376 targetElement="reco_cream_sugar"
                targetMethod="start"/>
      <bind test="/[@confidence $gt$ 10]"
        378 targetElement="drink" value="/drink"
        382 targetElement="confirm" targetMethod="start"
            targetElement="reco_yesno" targetMethod="start"/>
    </reco>
```

【図18】

フロントページの続き

(51)Int.Cl.[7]	識別記号	FI	(参考)
		G10L 3/00	571J

(72)発明者 ウァン クァンサン
アメリカ合衆国 98006 ワシントン州 ベルビュー サウスイースト 48 コート 16470

(72)発明者 ホン シャオーウェン
アメリカ合衆国 98006 ワシントン州 ベルビュー サウスイースト 58 プレイス 17797

Fターム(参考) 5D015 KK00 LL11